;login:

THE MAGAZINE OF USENIX & SAGE
October 2001 • Volume 26 • Number 6

inside:

USENIX & SAGE
The Advanced Computing Systems Association & The System Administrators Guild

USENIX Upcoming Events

5th Annual Linux Showcase and Conference

Co-sponsored by USENIX & Atlanta Linux Showcase, in cooperation with Linux International

NOVEMBER 5–10, 2001
Oakland, California, USA
http://www.linuxshowcase.org

XFree86 Technical Conference

(Co-located with the 5th Annual Linux Showcase & Conference)

NOVEMBER 8–9, 2001
Oakland, California, USA
http://www.usenix.org/events/xfree86/

15th Systems Administration Conference (LISA 2001)

Sponsored by USENIX & SAGE

DECEMBER 2–7, 2001
San Diego, California, USA
http://www.usenix.org/events/lisa2001

Conference on File and Storage Technologies (FAST)

Sponsored by USENIX, Co-sponsored by IEEE TCOS, in cooperation with ACM SIGOPS

JANUARY 28–29, 2002
Monterey, California, USA
http://www.usenix.org/events/fast/

Registration materials available: October, 2001
Camera-ready final papers due: November 15, 2001

BSDCon 2002

FEBRUARY 11–14, 2002
San Francisco, California, USA
http://www.usenix.org/events/bsdcon02/

Registration materials available: November, 2001
Camera-ready final papers due: December 4, 2001

The Fourth NordU/USENIX Conference (NordU/USENIX 2002)

Co-sponsored by EurOpen.SE and USENIX

FEBRUARY 18–22, 2002
Helsinki, Finland
http://www.nordu.org/NordU2002

Final papers due: December 7, 2001

The 3rd International SANE Conference

Co-sponsored by USENIX and the NLnet Foundation. Organized by NLUUG

MAY 27-31, 2002
Maastricht, The Netherlands
http://www.sane.nl

2002 USENIX Annual Technical Conference

JUNE 9–14, 2002
Monterey, California, USA
http://www.usenix.org/events/usenix02/

Freenix Submissions due: November 12, 2001
General Session Submissions due: November 19, 2001

2nd Java™ Virtual Machine Research and Technology Symposium (JVM '02)

AUGUST 1–2, 2002
San Francisco, California, USA
http://www.usenix.org/events/jvm02

Paper submissions due: February 4, 2002
Notification of acceptance: March 12, 2002
Camera-ready final papers due: May 28, 2002
Registration materials available: April, 2002

11th USENIX Security Symposium

AUGUST 5–9, 2002
San Francisco, California, USA
http://www.usenix.org/events/sec02

Paper Submissions due: January 28, 2002

2nd Workshop on Industrial Experiences with Systems Software (WIESS '02)

Sponsored by USENIX, co-sponsored by ACM SIGOPS, & IEEE TCOS

DECEMBER 8, 2002
Boston, Massachusetts, USA
http://www.usenix.org/events/wiess02

Submissions due: July 15, 2002

5th Symposium on Operating Systems Design and Implementation (OSDI)

Sponsored by USENIX, co-sponsored by ACM SIGOPS, & IEEE TCOS

DECEMBER 9-11, 2002
Boston, Massachusetts, USA
http://www,usenix.org/events/osdi02

Submissions due: May 24, 2002

For a complete list of future USENIX events, see **http://www.usenix.org/events**

contents

;login: Vol. 26 #6, October 2001

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

;login: (ISSN 1044-6397) is published bimonthly, plus July and November, by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $60 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.

Cover photo: Bill Cheswick at play at USENIX 2001

motd

Making Your Partner Be Right

by Rob Kolstad

Dr. Rob Kolstad has long served as editor of *;login:*. He is also head coach of the USENIX-sponsored USA Computing Olympiad.

kolstad@usenix.org

Over lunch at various conferences, my friend Dan Klein teaches me about "improv," the art of improvisational comedy. He tells me that one of the primary rules of improv is "Make your partner right." In the context of improv, this means that you accept your partner's lead and, hopefully in some constructive way, you amplify it. Imagine your partner suggesting that rain is imminent. Your response might be to open an (imaginary?) umbrella. This "makes your partner right."

I raise this topic because I think our industry (both the producers and administrators of software) exhibits very little of this "spirit of cooperation" that is demanded in the improv setting.

Consider this scenario: Your manager comes by to visit or invites you to a meeting. In the ensuing communication, it is learned that your institution is soon to merge with another institution. Ideally, this might lead to thoughts like:

"Oh boy, stable paychecks."
"Goodie; more bright people with whom to interact."
"Fabulous! Finally we will have sufficient resources to tackle the really big problems."

Regrettably, human nature and our culture being what it is, we're more likely to observe reactions like:

"Rats, more stupid managers."
"Oh no, now I'll be even more of a cog in a machine."
"These people have always been known to be bozos."

Now why is that?

I am absolutely sure that some of these sorts of negative reactions are cultural – both organizational and occupational. Organizationally, they were de rigeur in the past and are perpetuated as each new employee acquires the company or career culture. In the occupational context, I know that as a trained engineer I can spot problems a mile away. Unfortunately, I have as much trouble as anyone convincing management that the problems truly exist and that appropriate resources (funding) should be devoted to attacking them. So, I tend to react negatively when I don't think an idea's creator (or even announcer) has thought through the "big picture."

Is negativity just human nature? Is it a rational learned response/reaction to a long history of missed expectations in the engineering field?

Whatever it is, it is truly a pain if one believes that "vision" is important. Consider: "Let's have a conference on XXX." "Nah, no one will come." That sounds contrived, I know. Here's a real example I received an email last week from a 20 year old intern at a Large Computer Research Lab: He says in a meeting: "Let's go ahead and sort the ten PC prices in our program." They respond: "No, the database has optimized sorting algorithms. We don't want to rewrite the database. Besides, it will take a long time to sort the numbers."

Of course, sorting ten numbers is a sub-millisecond task. Amusingly, the note to me was entitled "I am working with idiots." As it turns out, he was, in fact, working with idiots, as near as I can figure.

Of course, those idiots had the same sort of cultural markers that we all seem to exhibit from time to time: "Let's not do it a new way, we know and understand the both the idiosyncrasies and complexities of the old way(s)." Sometimes, this is a valid response. In my friend's case, I think his mentors were a bit misinformed or inexperienced. (Another paragraph he wrote me notes that programs they write are supposed to avoid the use of constructors because ". . . it says right here in the textbook that constructors are slow.")

So I have a proposal for all of us. One of these days when the workload is no more stressful than usual and you feel like you have a slight excess of "good energy," give it a try. For that day, see if "making your partner righ" is a reasonable strategy. Support people's ideas with your own constructive suggestions. Encourage them to apply themselves most fully in their visions and ideas. See if you can help them with your own style and expertise.

Let me know how it goes – I'll publish the most interesting responses.

apropos

Haphazard Heroics

by Tina Darmohray

Tina Darmohray, co-editor of *;login:*, is a member of Stanford University's Network Security Team.She was a founding member of SAGE.

tmd@usenix.org

System administrators often lament that when they're doing a good job, no one knows it. As such, it's hard to get recognition in the workplace or raises at review time. This non-recognition situation is disturbing, but lately I've witnessed an opposite trend in the workplace, which is equally disturbing. Over and over again, I see system administrators succumb to the pressure to over-commit or under-plan, landing themselves and their co-workers in situations that require heroic efforts to dig themselves out. I think this can and should be avoided.

Over-committing doesn't help anyone. It may feel better in the short run to give a "can do" answer to the management, but if it's unrealistic, it's better to say so up front. Too often IT managers fall into this trap. Recently, a friend witnessed this dilemma. The company was outsourcing a mission critical application, but had become unhappy with the situation. A dead-of-the night scenario was concocted in which the company would request a current tape of the data as an "upper-management disaster preparedness drill" so as not to tip off the outsourcing company. Next, in a mere 48 hours, the IT group would bring up new RAID servers, install database software, and restore the tapes. If you're not gasping yet, let me add that the RAID servers hadn't been procured yet!

Of course, this became a fire drill for all involved. Just getting the machines onsite and running was a formidable task. Bringing them up taxed the existing machine room cooling capacity, so for days, the doors were propped open, the door alarm was temporarily disarmed, fans were brought in, and the IT group anxiously watched as the new machines teetered on the brink of overheating until the AC guys could get out and rework the system.

Meanwhile, contractors were brought in to do shift work to get the database up and the tapes restored. The round-the-clock shifts were uncomfortable in the over-heated machine room, but the oppressive heat paled in comparison to the pressure-cooker environment the entire IT group was working in, now that the entire company's productivity hinged on getting these machines up.

In the end, they didn't make the deadline. It eventually took about two weeks to get everything working, which would have probably been a reasonable "can do" estimate in the first place. In my book, the inability to size the project was a failure on the part of the IT manager, but she proclaimed victory via heroics instead. Somehow unrealistically sizing the job up-front was overlooked, and instead, pulling the all-nighters, and hiring contractors to do the same, was portrayed as "going above and beyond" to get the job done.

Often, unrealistic demands come down from above, and it's hard to say no to them. But some folks do it to themselves! Email upgrades are always fertile ground for such problems. Email is typically the most visible computing service, which means that email upgrades are prime candidates for maximum planning and minimum upheaval. Yet email upgrades go awry, often due to the folks doing the upgrading! Midway through the afternoon of one such upgrade involving migration to two new redundant mail gateways, the IT manager suggested the site also migrate the DNS servers in house: IP address change and all. "What about notifying the NIC?" I squealed in terror, but there was no stopping her. She argued that we'd co-locate the inconvenience and come out ahead for doing so. Not the case. The mail successfully rolled over to the awaiting servers with no problems. Then, we brought up the DNS servers in-house easily enough too. But, of course, the NIC changes lagged predictably behind, and we had to undo and redo the cutover again. I felt there was no excuse for such a mid-course change of plans, but she hailed herself as a hero, and I wondered how anyone could agree.

There's also the appearance of heroics by the chronic all-nighters. These are the folks who wander into the office mid afternoon on a daily basis, or skip a few days entirely and then come in for several days running. They're frequently the topic of water cooler conversations about their seeming dedication to the job; putting fire fighting above sleeping on a mattress, eating self-cooked meals, or taking a shower at home. But when I do the math, I'm not sure they're any more dedicated, on a straight hour by hour basis, than the guy that gets there fed, clean, and rested at 8 a.m. each day and spends all of his efforts on-site attending to the machines and network. Too often, the quality of work which comes out of the all-night heroes reflect the lack of sleep and lack of planning.

Fire fighting is part of our job. Hardware fails, systems wedge, and occasionally, there's a virus or two let loose on the network. Going into the blazes when it's necessary is a noble and reasonable thing to do. However, over committing, under planning, or creating your own crisis is not a good approach, and fighting those kinds of fires makes for hapless heroes at best.

;login:

EDITORIAL STAFF

EDITORS:
Tina Darmohray *tmd@usenix.org*
Rob Kolstad *kolstad@usenix.org*

STANDARDS REPORT EDITOR:
David Blackwood *dave@usenix.org*

MANAGING EDITOR:
Alain Hénon *ah@usenix.org*

COPY EDITOR:
Steve Gilmartin

TYPESETTER:
Festina Lente

PROOFREADER:
Lesley Kay

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES
USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: *office@usenix.org*
login@usenix.org
conference@usenix.org
WWW: *http://www.usenix.org*

letters to the editor

THE COST OF GOING TO CONFERENCES

from Art Mulder
amulder@irus.rri.ca

Dan Geer had an interesting column in the July 2001 issue of *;login:*.

I completely agree with his thoughts about pursuing excellence in your work, learning as much as you can, working hard. I also, like Dan, have found the USENIX association with it's publications and conferences, to be a great source for learning and networking. I'm already looking forward to attending the LISA conference in December.

Yet I was struck by his comment: "For much of my career, I have attended USENIX on my own nickel . . ."

My initial reaction was that I wasn't sure if this was realistic or even possible for most people. However, Dan goes on to say "the pain was more than compensated by the gain." I can't really argue with that, it's one person's opinion after all. Affordability is a judgement call also – once we decide something is important, we can usually find a way to afford it.

Still, I thought it would be a worthwhile exercise to do the math here, and share the results with you. Let's just see what Dan is suggesting here . . . (I've detailed my calculations below). I came up with a cost of approximately US$1650 for someone to attend the LISA conference this December. (Or about $2500 in Canadian funds, for someone like myself.) I find that to be a pretty steep price to pay.

Furthermore, I have to assume that if I'm paying for it myself, then it's also on my own time, so that's four days out of my annual vacation.

So, was Dan writing in his column in is official capacity as president of the USENIX association? Can we conclude that it is the official policy of the USENIX board that their conferences should be affordable enough for us to pay out of our own pockets? (OK maybe I'm being a bit facetious here)

I checked the 2000 USENIX salary survey, and if I read it correctly, the majority of USENIX members receive 3 or more days of training per year, paid by their company. Hopefully this is not an issue for most of us. I would be curious though to know specifically how many people do pay out of their own pocket to attend the USENIX conferences. Perhaps a more specific question on next years salary survey?

So what about myself, would I be willing to pay for this most excellent conference out of my own pocket? I think I am reasonably well paid, and I think I get a big benefit from attending, but 4 days out of my personal vacation time, plus $2500 out of my family budget? Sorry, No.

I think I can say in conclusion that I'm glad that my employer has a travel budget and sees the value of conferences for its employees.

My Calculations:

$120 – USENIX membership fee, might as well include it, as we'll need to pay it sometime during the year.
$510 – LISA technical sessions fee (Wed-Fri, Dec 5–7, 2001) (NOTE: adding a one-day tutorial would nearly double this)
$374 – hotel costs for Tue. – Sat. evening (The travelocity web site gives the conference hotel cost to be $747 incl. taxes) assuming that you share the room with one other person and that you stay in the conference hotel (which USENIX requests us to do since they would incur substantial penalties if they did not fill up the block of rooms arranged in their contracts. I further assume that this prevents them from informing us of cheaper hotels in the vicinity).

letters to the editor

I also assume that the average attendee will be flying in, and will, therefore, require a hotel for Tuesday-Saturday evening. (It had better be the majority, if this truly is a international conference) Note this requires at least 4 days off from work. (If you only earn two weeks of vacation, this is virtually 50% of your vacation allotment, if three weeks, this is still 25-30%)
$150 - food 6 days (Tue-Sun @ $25/day)
$100 - incidentals (taxi, bus, long distance calls, fudge factor)
$400 - Average flight cost: The travelocity Web site gave these flight costs – as of July 2001: Chi-San Diego = Average $330, NY-San Diego = avg $400, Denver-San Diego = avg $440, Atlanta-San Diego = avg $393. And the Air Canada Web site gave these prices : Toronto-San Diego = $CA 840 (US$540) Vancouver-San Diego = $CA 700 (US$ 455)

$1654 – Total US Dollars
($2544 Canadian Funds)

Dan Geer comments:

Of course, only a tiny percentage of USENIX attendees pay their own fees; most are covered by their employers. I think that education really is less expensive than ignorance and wanted to emphasize that thought.

AN EXCHANGE ON CHARITY

from Anne Bennett
anne@alcor.concordia.ca
I just read Andrew Hume's note "What's Up with Charity" in the June 2001 *;login:*. I was pleased to see USENIX getting involved in providing and setting up computer equipment for poor people, even though so far the efforts (or at least those he reported) have been centered in the States.

It's easy for many of us to forget that there is very much a "digital divide" between the rich and the poor (this is true not only within first world countries such as the USA and Canada, but even more so on a planetary scale). Yet, we computer professionals are very well placed to understand the huge negative effect of not having access to the net on someone's ability to function in society – and it seems a particularly appropriate form of charity for us to try to remedy this problem, even in small ways. I'd be very interested in seeing the results of those projects published, so that others who also want to help can learn from our successes and failures.

I'd like to see this type of project continue to be funded by USENIX.

Dan Geer Replies:

Thank you for your comments.

Might I ask you in return if you think it better to overcharge for the services USENIX offers and then to put that money to unrelated charitable work or to undercharge so as to permit any charity to be a decision left to the individual member as to what charities they wish to support and how, a decision then unbuffered by the personal whims of a simple majority of the USENIX board? This money is not free money – it is your money and when any middle man, USENIX being no exception, handles monies said monies shrink. This is true of taxation; it is true of the United Way; it is an economic reality.

You doubtless did not wish to enter debate, and you are clearly welcome to demur further conversation. Your note is one of a tiny few received in any case, which thus makes it special.

and Anne Bennett replies:

It's true that I'm not really yearning to debate this at length, but your points are relevant, and they are certainly issues that I have considered. In the general case, I'd agree with you – for example, if USENIX proposed donating to some random charity, even one I agree with, I would not be in favor, for exactly the reasons you raise. In fact, despite my support for the EFF's work, for example, I'm not particularly in favor of USENIX donating funds to it (though I would not jump up and down to prevent it, either). Supporting that cause with expertise in our area is another matter.

However, the examples given by Andrew Hume seemed to me quite different, in this respect: they address needs which the "general charitable community" is not really yet equipped to handle. It is not only difficult to find and to evaluate the expertise needed to "put the poor on the net," but more importantly I think, the need to do this is not yet properly recognized by the general population. This is an area in which we computing professionals are in the best position to kick-start the action; in fact, as fans of UNIX and as proponents of open-source software (which I suspect a good number of us are), we would do well to seize the opportunity to show that open-source operating systems on older machines can serve such needs at a fraction of the cost of commercial systems.

In any case, whether or not we are able to use this kind of charitable opportunity to advance the cause of our operating systems of choice, it still seems to me that, until the general population realizes that access to the Internet is about to rival access to affordable education with respect to each individual's full functioning in society (or until the digital divide disappears, hah!), it is very appropriate for us of USENIX to get involved in solving this problem.

So yes, I'd like to be overcharged for my USENIX membership (which to me is not only a question of "member services") in order to permit us to support charitable work which I consider far from unrelated to our goals. The "innovation and research that works" which we are supposed to be fostering has to "work" in society, not just in the lab.

conference reports

This issue's reports are on the13th Annual Computer Security Incident Handling Conference (FIRST) and on the USENIX 2001 Annual Technical Conference.

OUR THANKS TO THE SUMMARIZERS:
for FIRST:
Anne Bennett
for USENIX 2001:
Josh Simon, Coordinator extraordinary
Brandon Ching
Peter da Silva
William R. Dieter
Kartik Gopalan
Chris Hayner
Adam Hupp
Rosemarie Nickles
Joseph Spadavecchia
Jon Stoffel
Kenneth G. Yocum

13th Annual Computer Security Incident Handling Conference (FIRST)

TOULOUSE, FRANCE

JUNE 17–22, 2001

Summarized by Anne Bennett

The Forum of Incident Response and Security Teams (FIRST) is a global organization whose aim is to facilitate the sharing of security-related information and to foster cooperation in the effective prevention and detection of, and recovery from, computer security incidents. It holds several technical colloquia each year which are open to members only, and one annual conference which is open to all.

TUTORIALS

LEGAL AND OPERATIONAL ISSUES AFFECTING EVIDENCE PRESERVATION AND RECOVERY IN INTRUSION CASES

Byron Collie, Wells Fargo Services Company, USA; Steve Romig, Ohio State University, USA

Computer forensics is defined as the process of identifying, preserving, analyzing, and presenting digital evidence in a manner that is acceptable in legal proceedings. A number of computer and network intrusions (and other crimes which make use of computers in some way) cannot be successfully prosecuted because the evidence has been lost, destroyed, or mishandled.

Planning for correct incident handling includes not only acquiring relevant tools and making sure that staff know how to use them, but, possibly more importantly, putting in place the organizational structure which will make it possible for people to act quickly: for example, identifying who has the authority to release log information, start network monitoring, or make the decision to investigate or to prosecute.

Computer evidence is volatile and fragile; as soon as an incident is suspected, immediate action must be taken to preserve the evidence.

It can be hard to decide whether it is best to shut down the computer gracefully (risking booby traps which may have been inserted into the shutdown sequence), just kill the power (risking losing valuable data as well as compromising the integrity of the file systems), unplug it from the net (risking a possible "dead man switch" inserted by the attacker which could delete all data if the network becomes unreachable), or leave it running (risking further damage or liability to other parties). Just don't reboot: that's the worst choice of all, as it is likely that the intruder has installed programs that will start at boot time, and /tmp will be cleared and other information may be overwritten.

If possible, acquire the volatile evidence, such as the list of open network connections (with netstat), the process list (ps), the list of open files (lsof), and so on. But at the same time, be aware that everything you do risks destroying evidence – for example, by overwriting parts of memory or the swapfiles. Make sure you document everything you do, but not on the compromised system! A tape recorder may be helpful at this point. One critical piece of volatile evidence is the clock drift (difference between system time and "real" time), without which it may be impossible to later correlate timestamped log entries from various sources. Do *not* change the clock!

Also, take copies and MD5 (or better, SHA-!) checksums of relevant data, checking that the checksums of the original and copy match. Sign and date these checksums, possibly using a digital time-stamping service, and place evidence, checksums, and signatures under lock and key. With respect to what to copy, a bitwise copy of the entire hard disk is best, followed by a bitwise copy of the file systems, followed by copies of files.

File extracts are unlikely to stand up in court.

Log extracts may be helpful when *presenting* evidence, but complete records must be submitted to the court; some courts will accept log files in digital form (CD-ROM), while others will insist on inches of printout! Note in particular that IDS logs are incomplete, and while an IDS is a great burglar alarm, better and more complete sources of evidence exist in the form of system logs, router and terminal server logs, etc.

Once you have documented the scene (i.e., noted any users at the computer, which switch port the host is plugged into, etc.), acquired whatever volatile information you can, made copies of the disks, and taken whatever actions are necessary to exercise due diligence with respect to your liability (e.g., protected your information and services, as well as any third parties that may be involved), it's time to analyze the data.

Since you'll be analyzing copies of the data, store the original data (disk images from compromised hosts, router logs, terminal server logs, etc.) in a safe place to ensure that you preserve the continuity of the evidence (i.e., you protect it from tampering). Data analysis requires a reasonably deep understanding of how evidence is created, what might be missing, and what can go wrong. For example, a particular entry in a UNIX wtmp file might indeed correspond to a user login, but it might also have been faked by an intruder, and even if not, there's no guarantee that the account owner was the one who logged in.

When correlating logs from various sources using timestamps, be aware of (and correct for) clock drift and time-zone disparities. Also, be aware that some activities are logged when they begin (such as tcp_wrappers logging the start of a Telnet session) and some when they end (wtmp contains the time when a login successfully completed). And, of course, take into account that the logs themselves may have undergone tampering by the attacker: they may have been overwritten, or the software that produces log entries may have been modified. Syslog-type logs sent over the net use UDP and are subject to data loss and spoofing. Guard against these problems by using data from as many different sources as you can. If you've been logging to a secure log server, all the better.

As for analyzing the contents of a disk, be aware that many tools exist to help you reconstruct deleted files: Farmer and Venema's Coroner's Toolkit, for example. You'll be looking for the "standard" stuff, such as specific text fragments pertaining to your incident, IP addresses, email messages, and so on. Some people have found it useful to build a Web-style index on the files on disk and search that!

Don't neglect the backup tapes (you protected them, right?); they can show you when and how certain files changed. File checksums (such as Tripwire) are an invaluable aid – especially if you have checksums of the known "clean" system or of a virgin system — but if not, it's still useful to compare data from backups.

Investigating Malware Incidents

Christine M. Orshesky, i-secure Corporation, USA

A large majority of sites seem to be making some use of antivirus software, yet in recent surveys, well over half had suffered malware infections. Some of this is explained by incorrect use or infrequent updates of the AV software, but one must remember that there will always be a time lag between the launch of a new virus or worm and the availability of countermeasures from the AV software vendors.

Malware was defined as software, firmware, or hardware that is intentionally introduced into a computer system for unauthorized purposes, usually without the knowledge or consent of the user. This session covered software instances only. Note that the malware may or may not inflict actual damage.

Malware was classified as:

- Viruses: attached to an existing file, such as a diskette boot sector (boot sector virus), a program file (file infector virus), or a document (macro virus).
- Worms: self-contained programs which spread from system to system, usually over the network, often using the email system to propagate themselves.
- Trojans: programs which masquerade as a legitimate program to trick the user into invoking them.
- Hoaxes: malware warnings which count on human intervention to re-mail them to large numbers of people. While they do no direct damage, the resulting volume of email traffic can cause problems. (Summarizer's note: we have recently seen "virus warnings" which trick the receiver into manually deleting legitimate files!)
- Logic bombs: unauthorized code introduced by the programmer of an application, which performs some action based on a trigger event. For example, upon finding that the author is no longer on the company's payroll, they might destroy all of the business records.
- Nasty or joke programs: (no definition).

Many tools, including the well-known anti-virus programs, are available to help combat malware. These range from scanners (which look for known attacks) to heuristics-based behavior checkers (which can detect unknown attacks but suffer from false positives) to file

Malware Name	Year Created	Time to Become Prevalent	Type	Cost of Damage
form	1990	3 years	boot sector virus	$50M over 5 yrs
concept	1995	4 months	Word macro virus	$50M
Melissa	1999	4 days	email-enabled Word macro	$93M to $385M
LoveLetter	2000	5 hours	email enabled script	$700M

Changes over time in malware

integrity checkers (which can report unwanted changes but not how they happened). In addition, firewalls and router filters can block access to known problematic sites or traffic types (to stop the delivery of viruses), intrusion detection systems monitor the network for signs of compromised computers, and content filters are used for files downloaded from the Web.

HOT TOPICS

S/MIME Integration in SYMPA Mailing List Management Software

Olivier Salaün, CRU – Université de Rennes I, France

SYMPA is a mailing list management program which is available under the GPL license; it is used by about 2000 sites. Release 3.0 features:

- Authentication for submission based on S/MIME signatures of messages.
- Encryption for outgoing messages: decrypt once upon receipt with the list key, encrypt outgoing messages for each recipient with each recipient's key.

SYMPA itself uses an RDBMS to store mailing-list information, to ensure performance and scalability. It has a Web interface for both subscribers and list owners; it has a shared document repository; and it has been localized to several languages.

OpenSSL is used to implement certificates, and user X509 certificates are used not only in the S/MIME messages but also for authentication in Web interactions. Note that CRLs (certificate revocation lists) are not yet implemented.

Authentication customization is available per list, per command; it defines who (subscriber, list owner) may do what (subscribe, review, send), and the authentication methods accepted (SMTP, MD5, S/MIME). Note that PGP is *not* implemented.

For more information: *http://www.sympa.org/*.

SRMail (Secure Remailer)

Cory Cohen, CERT-CC, USA

Goal: to address the needs of the FIRST community mailing lists, and to avoid sharing a quarterly changed symmetric encryption key.

Problems:

- Many incompatible mail encryption methods (try to support as many as possible)
- Changing technologies (e.g., many versions of PGP, which keeps evolving)
- Implementation incompatibilities (e.g., different PGP packet formats).
- Key management problems
- MIME is not universally adopted, but is desirable. Moving to it while maintaining backward compatibility is hard
- Scalability problems: must be able to send 1000 differently encrypted messages
- Security concerns: implementation must be solid and reliable.

SRMail can:

- Send signed and encrypted form letters.
- Decrypt encrypted data, verify signatures.
- Manage contact information and encryption keys (associate encryption keys with organizations).
- Manage access to encryption keys, especially shared keys.
- Manage an encrypted mailing list: the sender signs the message and encrypts it with the mail-server key, the server decrypts the message and validates the signature, then it re-encrypts and signs the message for each recipient.

IPv6 Migration and Security

Jean-Jacques Bernard-Gundol, Hervé Schauer, Consultants, France

IPv6 is the next-generation Internet protocol; it has new features and new security issues. The IETF has proposed several possible migration methods from the current IPv4 to IPv6, and some of these are vulnerable to problems. For example, if we tunnel IPv4 inside IPv6, then someone can insert a bad IPv4 address into the inner packet, and the IPv6 stack will blithely unpack and use that address; similarly, tunneling may bypass "usual" checks. NAT-style solutions (with protocol translation) are quite vulnerable to denial of service. IPv6 supports multi-homing in a way that may make ingress/egress filtering much more difficult.

Authorization and Privacy of Internet Applications

Yves Deswarte, LAAS-CNRS, France

Good security can breed a need for better privacy; DDoS, e-commerce fraud, and transnational crime have spurred the development of better security practices, including more reports and monitoring. It has become easier to collect private information, which, while it may enjoy legal protection, is rarely protected in practice. There's no economic pressure for privacy; security research is funded by security agencies, not by civil libertarians!

Client-server implementations decide to grant or deny access based on identity, so information about people's identities is collected. Many transactions now involve more than two parties, such as: a customer, a merchant, a bank, a credit card company, etc. Most of these parties do not *need* very much information about the others, but they often collect it anyway; for example, the merchant should not need to know *who* the client is, but only if the payment is OK.

Of course, in the case of a judiciary request (e.g., to prevent money laundering), it should be possible to disclose real identities. The proposed solution involves a set of central authorization servers, each of which holds only part of the information identifying the parties in a transaction. According to this mechanism, only a judge can get enough data to actually decode the information and reveal those identities.

KEYNOTE

Ensure Security and Confidence in Cyberspace, a Priority for France

Henri Serres, Secrétariat Général de la Défense Nationale / Service Central de la Sécurité des Systèmes d'Information (Service du Premier Ministre), France

The French approach to information security was described. France, like many other countries, is undergoing rapid development of electronic technology and is establishing itself as a major player in cyberspace. The government action program for a cyber society has as its goals: (1) to connect everyone, avoiding a "digital divide"; (2) to support French commercial involvement in this new economy; and (3) to improve security and increase confidence in cyberspace.

Protecting the infrastructure and ensuring safe and honest transactions are a strong priority for the French government. French legislation has kept up with new crimes such as malware and unauthorized access. The French government has intervened at the level of personal data protection, implementing the European directives in that area, has recognized the legal value of digital signatures (a decree sets up rules for certificate and signing authorities, again compatible with the European approach), and has fully liberalized its encryption laws, without key length restrictions.

In addition, government enforcement has been strengthened: a central office for high-tech crimes now exists to assist other forces; CERT-A has been created to assist government bodies with computer-related incidents and attacks; a Central Directorate for Information Security now reports to the prime minister and has a role as a national regulatory authority, monitoring, for example, cryptography products for government use (and also the French scheme for certification). The directorate also participates in operational matters, assisting government departments in setting up their network infrastructures.

CSIRT OPERATIONS

Incident Organization and Security Incident Handling

Jimmy Arvidsson, Telia AB HQ – Telia CERT, Sweden

A taxonomy of events was established: event, incident, security incident, crisis, catastrophe. When there is indication of activity, the activity should be categorized into this taxonomy. Appropriate actions can then be identified. Events can be handled if necessary, recovered from, legal action taken if appropriate, and then the whole event can be followed up on to improve procedures.

The author suggests a first level of assessment, where the type and severity of the incident are determined, and an "incident owner" is contacted; the incident owner would be a representative of the entity responsible for the systems affected: for example, a departmental manager, or the owner of the host or information affected. Also at this initial stage, "first aid" might be applied as necessary; for example, in the case of a Web server defacement, the system might be taken "offline" using the DNS.

Then, a second level of assessment is performed. "Events" are merely logged. "Incidents" are handled by permanent operations staff. "Security incidents" might merit the setting up of a virtual CSIRT: a temporary, project-oriented response team whose existence ends with the resolution of the security incident. A "crisis" or a "catastrophe" would be referred to a crisis management team.

The virtual CSIRT draws on existing resources and competence, and can be especially useful when the size or budget of the organization makes it difficult to justify a permanent CSIRT. A security manager might take the role of incident leader, and CSIRT members might be recruited from three groups of people: the incident owner (system owner, departmental manager, information owner), specialists (sysadmins, network admins, central CERT), and administrative people (help desk, lawyers, public relations).

Introducing Constructive Vulnerability Disclosures

Marko Laakso, University of Oulu – OUSPG, Finland

The author is looking for a compromise between full disclosure and non-disclosure models for software vulnerabilities; he proposes "constructive vulnerability disclosure." The PROTOS project, of which he is a member, studies methods to test protocol implementations for security vulnerabilities.

Consumers continue to be plagued by computer vulnerabilities, many of them avoidable. Poor software quality (leading to large numbers of vulnerabilities based on known trivial programming errors), the inefficiency of the traditional vulnerability reporting/fix/release process (reappearance of vulnerabilities in future releases, small variations which bite multiple vendors, inability of customers to evaluate products), and waste of time (the time used debating full/non-disclosure would be better spent addressing the real issues!) continue to impede meaningful progress in this area.

The goals of the PROTOS project are:

- Low-cost black-box evaluation of products
- Early elimination of some of the most trivial vulnerabilities
- Vendor awareness beyond one particular vulnerability
- Regression testing of future versions
- Customer-driven product evaluation

The author's group created software to bang away at products and report problems; the results are packaged and released initially to vendors, though with the identities of the competitors removed. After a pre-announced grace period, the test suite is released to the public.

Among the first fruits of the project were a test suite for WAP, the Wireless Application Protocols suite, which generated 4236 test cases and tested seven WAP gateway products. All implementations failed at least some of the tests; some implementations failed in half of the 39 test groups. The results were reported to the vendors, along with, privately to each vendor, exploits (DoS in three cases, arbitrary code execution in the other four cases). Vendors had a grace period of at least 51 days before public disclosure of the test suite, and the entire process took 86 days. Vendor responses ranged from absolute inaction (in two cases) to prompt patches and advisories; a few vendors were even motivated to review their code more thoroughly.

For more information on the PROTOS project and its collection of test suites, please visit *http://www.ee.oulu.fi/research/ouspg/protos/*.

(Summarizer's note: a few weeks after the conference presentation, CERT Advisory CA-2001-18, "Multiple Vulnerabilities in Several Implementations of the Lightweight Directory Access Protocol (LDAP)," was released, which listed multiple vulnerabilities in nine different LDAP products, again based on test suites created by the PROTOS project.)

Experience with Abuse Management in Privacy-Enhancing Systems

David Bratzer, Zero-Knowledge Systems Inc., Canada

Zero Knowledge's "Freedom Network" provides anonymous Web browsing and chatting, and pseudonymous email and Netnews services. They tried to design their service to be resistant to abuse, recognizing that it is not possible to prevent abuse completely. They claim a 0.2% abuse rate, or one problematic account in 500.

DENIAL OF SERVICE

DoS Attacks on Transit Networks

David Harmelin, DANTE, UK

DDoS attacks via network flooding were studied. Usually, a master controller sends commands to a number of "handlers," which may in turn contact many compromised hosts to make them run denial-of-service software. The traffic tends to become aggregated in the transit network.

Each router in the transit network logs netflows to a workstation nearby. DANTE wrote a tool which has a central workstation "poll" each of these log hosts every 15 minutes and take a sample of 1/500 of the flows that occurred in a 10-second interval. For each router, an alarm is raised if there are more than 10 flows with the same destination IP address per sample.

About 98% of alarms were confirmed as attacks in progress; the tool can detect attacks at rates greater or equal to 100 packets/second. DANTE found that 90% of the attacks were "C class," i.e., were from a set of spoofed addresses all within the same class C network, to get through the ISPs egress filters. Most attacks (58%) lasted less than 15 minutes.

CSIRT COOPERATION

Collaboration of European Computer Security Incident Response Teams

Gorazd Bozic, Arnes SI-CERT, Slovenia

In the 1990s, an attempt was made to create a "EuroCERT" to coordinate interactions between European CERTs. This was the SIRCE project (1997–1999), which ended with the conclusion that a top-down approach to this task was not suitable.

In May 2000, TERENA established a task force to work with a wider number of individual CSIRTs. Why this in the face of previous failure? This time, a very informal process is being used, with quarterly two-day meetings. Here are some of the initiatives of TF-CSIRT:

- Trusted introducer program (signing PGP keys to identify CSIRTs to other CSIRTs)
- IODEF (Incident Object Description and Exchange Format) – workshops for new staff of CSIRTs
- Cooperation with EU officials (eEurope program) – clearinghouse for CSIRT tools

PROACTIVE CSIRT TOOLS

Experiences with National Wide-Scan Detect Systems

Hyunwoo Lee, Korea Information Security Agency, Korea

In 1999 the author's group experienced a set of automated, stealthy, distributed DoS attacks and, surprised by the scale of the attacks, felt that proactive countermeasures should be deployed immediately.

They found that when they received an alert about a DDoS attack, it was already too late to intervene. Traditional "passive" responses are ineffective and fail to stem the flow of attacks – manual email response is much too slow, and attackers have nothing to fear from the CSIRT community.

Port scanning is the initial step in attack preparation, so automatically detecting and reacting to port scans could help – though it is necessary to share that information with other members of the security community.

Realtime scan detection agent software was deployed to run at various end sites and report to a central data collector. Also, a system of alerts was developed within the community, where known incidents are reported as formal alerts, and suspicious occurrences as informal alerts.

It became possible to collect statistics of scan incidents and, using them, to detect and identify previously unknown attacks – for example, a sudden increase in scans on port 12345 was found to correspond to the Detlog worm. But the greatest gain of this project was the formation of a cooperative security community.

The CyberAbuse Project

Philippe Bourcier, XP Conseil, France

The CyberAbuse project developed from some IRC Undernet projects to prevent IRC abuse, especially DoS. The first such project, "Abuse-DoS," found "smurf amplifier" routers and sent information to administrators about the correct configuration of routers in this respect. Of the 190K misconfigured networks found, 25% were fixed after the project sent mail to the admins.

Another project, "Abuse-Proxy," addresses the problem of open IRC proxies, which provide anonymous IRC connections. The proxy scanner detects open proxies, and email is again sent to the admin of the misconfigured host.

In the "anti-hack" project, certain IRC channels were monitored for DoS tools and Trojans. Admins of victim sites, as well as CERT, were warned when compromised hosts were found. Problems were fixed by admins 80% of the time.

The CrimeWatch project makes available to security professionals and law enforcement agencies information about criminal groups and activities as well as new techniques.

Automated Incident Report Processing and Cross-Correlation of Probe and Scan Information

Mark McPherson, University of Queensland, Australia

CSIRTs receive numerous reports of many kinds of attacks, such as probes/scans, access, compromises, DoS, viruses, and spam.

Probes can indicate preparation for attack, or they may be a cover for some other attack in progress. The sheer number of scans provides a smokescreen which makes it quite hard to see what's really going on. Cross-correlation of logs across multiple sites can help pinpoint the "real" attacks; CSIRTs are the logical choice for collecting those logs, since they have already established trust relationships with their communities.

AusCERT created "probelogger" to collect, process, and acknowledge probe reports sent in by various sites. There is even a function to optionally report the scan to the originating site. If the origin of the probe is an AusCERT member site, the software raises a flag so that the incident receives special handling.

INTRUSION DETECTION

A Protection Mechanism for an Intrusion Detection System

Takefumi Onabuta, Information-Technology Promotion Agency, Japan

If the host on which the IDS is running suffers a system-level compromise, it is impossible to protect the IDS files and processes from the attacker. Thus, a kernel-level approach was taken, where mandatory access controls are implemented.

An access-control model (LOMAC) was considered which defines low- and high-security levels, and assigns these levels to subjects (processes) and objects (files); a low-level subject is prohibited from accessing a high-level object. Problem: system logs are written by low-level (userland) processes but read by high-level (IDS) processes — thus the information in the logs is not protected.

Another access-control model (LAM) was considered which defines different limitations on access to objects: read-only, write, append, create, delete, link, modify, execute.

The authors created a hybrid of LAM and LOMAC, called E-LOMAC (extended LOMAC), which not only permits access to high-level objects by low-level subjects, but limits even high-level access to specific operations. The new access-control system was tested on a host running a host-based IDS; most attacks were stopped. The system was benchmarked for performance, and it was shown that the impact was minimal (98.32% and 85.20% of no-E-LOMAC performance). E-LOMAC seems to successfully protect a host-based IDS from being disabled by an attack, but it is fairly difficult to configure.

SECURE PRACTICES

Securing Web-Based Applications with Hole-in-the-Chroot

Anne Bennett, Concordia University, Canada

A scheme was presented whereby it is possible to run a Web server and CGI scripts in a UNIX "chroot" environment and yet communicate with applications outside the chroot using files and named pipes. A daemon running on the "system" side listens for requests from the "chroot" side by monitoring a named pipe. When a request is received, it is checked against a list of defined job names, and the incoming data is checked before being passed to (possibly fragile) applications on the system side.

OS Fingerprinting

Franck Veysset, INTRANODE, France

Techniques for discovering the OS and version of a system on the Net were presented. They ranged from grabbing banners (Telnet greeting, HTTP header), to analyzing executables if such could be obtained (such as /bin/ls from an anonymous ftp server), to observing the behavior of the system's TCP/IP stack, especially in response to malformed packets but also with respect to TCP sequence numbers.

Panel Discussion: Ask the Experts

Moderated by Roger Safian, Northwestern University, USA

Q: Monoculture on the desktop (Microsoft) has caused a rash of problems; will monoculture on the Net (Cisco) do the same?
A: Juniper is starting to take some market share from Cisco. Within an organization, one must weigh the risks of such monoculture with the cost benefits in terms of ease of administration.

Q: What's the biggest bang for the buck in terms of securing computers and networks?
A:

- Assigning responsibility for keeping things up-to-date
- Sending people to conferences to establish networks of people who've "been there, done that" and who could be asked for advice or validation in difficult circumstances
- Assessing risks of highest impact; assigning someone to take the time to follow the security announcement mailing lists
- Use of digital signatures recommended to assist in detecting intrusions and recovering quickly from them

Q: What about kernel-mode rootkits?
A: They are out there and are quite effective, and their installation by crackers (including via worms) is increasingly smooth. Beware: NetFlow and traffic analysis are likely to detect them using unusual ports.

Q: If you had sensors sampling information about a network, what would be the most useful piece of information to have?
A: Analysis of flows before and during an event is the best; note that many IDSes do not provide that level of detail. Network flows are the best tool; they can be pumped through MRTG to see trends.

Q: How are the areas of incident response and viruses converging?
A: Worms combine the two; we see more and more overlap between viruses/intrusions and the use of the network. More cross-pollination is needed between the IDS and anti-virus industries.

POST-MORTEM ANALYSIS

Disk Analysis Hurdles

Philippe Bourgeois, CERT-IST – Alcatel, France

Disk analysis is sometimes required during an investigation of a compromised system or a legally seized system; this task is becoming more popular, and tools (such as The Coroner's Toolkit) are becoming available. Still, many things can go wrong or cause problems.

You may have trouble getting the disk image without cooperation from a sysadmin; you may have to bypass the BIOS protection to boot from alternative media (it would be dangerous to boot from a "hostile" system), or just move the disk to another host if that is possible.

Sometimes the file system is unreadable; you'll have to use data recovery tools to try to reconstitute files from blocks of data on the disk.

Dealing with large disks can be a problem, and disks are getting larger all the time. To reduce the forensic effort:

- Focus the investigation on a specific set of files.
- Discard from consideration all "known good files," based on MD5 signatures of the OS and applications, and analyze only unknown files. This can easily remove most files from consideration.
- Try indexing the data on disk to speed up searches.

When faced with encrypted data, check for weak encryptions which are easy to break. If necessary, try a brute-force approach to guess the key. However, be aware that these efforts may well fail. Don't forget that the plain text may be somewhere on disk as a deleted file or part of a swapped-out process.

INDESTRUCTIBLE INFORMATION

Wietse Venema, IBM, USA

Although commonly received wisdom suggests that it is very hard to recover a deleted file (since its blocks are reallocated and often overwritten), it turns out that data on disk can be read, assuming appropriate equipment, even after having been overwritten several times.

Sorting files (including reconstituted files) by time (access, modify, create) can often show what happened on a system; for example, access to compilers, libraries, and header files shows a compilation. Of course, bear in mind that file times can be forged! Linux rootkit v4 has a "footprint" of about 800 file changes, of which about 460 are deleted files (probably rootkit source).

In practice, the longevity of deleted files can be quite significant; a 10-month-old machine was examined, and numbers of deleted files (by age in one-month increments) ranged from 172 at four months to 51205 files at 10 months. In one case, a compromised Linux honeypot was examined, but traces of its previous lives running Solaris (including a firewall config file!) and Windows 95 were found in unused space.

USENIX 2001 Annual Technical Conference

BOSTON, MASSACHUSETTS

JUNE25–30, 2001

INTRODUCTORY REMARKS AND KEYNOTE ADDRESS

Summarized by Josh Simon

INTRODUCTORY REMARKS

The conference began with Dan Geer, the president of the USENIX Association, thanking Clem Cole and Yoonho Park for their work in putting the conference together.

Following the usual general announcements, the Best Paper awards were presented:

- General Track – Best Papers were awarded to "A Toolkit for User Level File Systems," by David Mazières, and "Virtualizing I/O Devices on VMware . . .," by Jeremy Sugerman et al.
- Freenix Track – Best Paper went to "Nickle," by Bart Massey and Keith Packard; Best Student Paper went to "MEF: Malicious Email Filter," by Matthew Schultz et al.

Following this, USENIX Vice President Andrew Hume presented the USENIX Lifetime Achievement Award (also known as the "Flame") to the GNU Project. Andrew then presented the Software Tools User Group (STUG) Award to the Kerberos development team for its secure, scalable, and relatively simple-to-administer suite of tools. Ted T'so accepted on behalf of the team and donated the $1,000 cash award to USENIX to be used for student stipends for August's USENIX Security Symposium. (See *http://www.usenix.org/directory/awards.html* and *http://www.usenix.org/directory/stug.html* for details.)

KEYNOTE ADDRESS

Dan Frye, director of IBM's Linux Technology Center, spoke about Linux as a disruptive technology. The term isn't intended to have any derogatory connotations; rather, the talk focused on how the growth of Linux has disrupted the status quo of how businesses choose IT products. This year alone IBM is pouring $1 billion into Linux development, working within the public development community, because of business decisions (Linux makes money for the company and for the shareholders) instead of purely technical ones.

A disruptive technology is one where the skills, the desire, and an open culture of significant size all meet. The desire for Linux and the openness of the community are well documented. Further, over time, the skills in computing have moved from mainly academia (meaning colleges and universities) to all levels of education, as well as to hobbyists and even industry, thanks in part to the explosion of games, the Web, the growth in technology, and so on. The increasing commoditization of technology has also fueled the explosion of skills.

IBM believes that Linux as a technology is sustainable in the long term. It's got growing marketplace acceptance, it doesn't lock the customer into a particular vendor for hardware or software, is industry-wide, runs on multiple platforms, and is a basis of innovation. Linux has become critically important for e-business due to the confluence of desire, skills, and the open culture with an ever-growing community size.

Dr. Frye went on to dispel some rumors about Linux in the enterprise environment:

Myth: Open source is undisciplined.
Fact: The community is very disciplined, reviewing code and assignments and making sure things are "right" before rolling them into a major distribution.

Myth: Open source is less secure.
Fact: Because of public review and comment to prevent security holes from getting released (or from staying in released code unpatched for long), open source is as or more secure.

Myth: The community doesn't do enterprise features.
Fact: The community wants good designs, but it is not against enterprise features. Designing good, scalable solutions – whether for multiple processors (threading code) or different architectures or clusters, or backing up over high-speed devices (networks) – is a major goal of the community.

Myth: The open source community will fragment.
Fact: Although such fragmentation is possible, IBM believes it is unlikely.

Myth: Traditional vendors cannot participate.
Fact: Untrue; IBM is competing quite well, and other vendors such as Compaq and Dell are making open source OSes available on their hardware platforms as an option for customers.

Myth: Open source doesn't scale.
Fact: Definitely untrue. Open source works on the enterprise scale and in clustering environments.

Myth: Open source has no applications for it.
Fact: Open source has over 2,300 business applications, not counting the many non-business applications that run under the various versions of Linux and *BSD.

Myth: Open source is only a niche market.
Fact: Open source OSes are on nearly a quarter of the servers in production.

Myth: Open source is never used in mission-critical applications.
Fact: While this may have been true in the past, it's becoming less and less so. It should be mission-critical-capable in the very near term.

The IBM Linux Technical Center's mission is to help make Linux better, working within the community. Their URL is *http://oss.software.ibm.com/developerworks/opensourcelinux.*

FREENIX Track

MAC Security

Summarized by Adam Hupp

LOMAC: MAC You Can Live With

Timothy Fraser, NAI Labs

Despite proven usefulness, MAC security models have not been widely accepted. Their primary obstacle has been a high cost of use caused by incompatibility with existing applications and users. LOMAC attempts to solve these problems by implementing MAC security that is transparent to most users, does not require site specific configuration, and is compatible with existing applications. LOMAC uses the Low Water Mark model of protection, which applies well to UNIX systems. It partitions both files and processes into high and low levels. The high level contains critical portions of the system such as init, libraries, and configuration files. The low level is made up of all other system processes and files. Once a high-level process accesses a low-level file it will be demoted. Additionally, low-level processes are unable to signal high-level processes or modify high-level files. This prevents a potentially compromised process from affecting other areas of the system.

LOMAC uses a simple static map to determine the level of files. For instance, all files under /home will be low level, while /usr/sbin would be at a high level. In some cases a program (such as syslogd) must access untrusted resources and still modify high-level files. In these cases the system allows exceptions in order to maintain compatibility. In benchmarks LOMAC had a small performance penalty of between 0–15%. To give a good example of LOMAC's transparency, an experienced user had it installed for 11 days without realizing that it was there.

More information is available at *ftp://ftp.tislabs.com/pub/lomac.*

TrustedBSD: Adding Trusted Operating System Features to FreeBSD

Robert N. M. Watson, FreeBSD Project, NAI Labs

Implementing trusted operating system features can significantly enhance the security of a system. The TrustedBSD Project aims to integrate several new features into FreeBSD that improve security and ease future development work. The most visible new features are MACs, ACLs, and fine-grained privileges. Some of the features such as ACLs and MACs are scheduled to be released in the upcoming FreeBSD 5 kernel. Equally important are auditing, cleaner abstractions, and documentation. The TrustedBSD team found that security checks were often implemented differently in different areas of the kernel, which can lead to bugs.

Watson discussed some of the lessons they had learned in the development process. They found that it was much more effective to work closely with the main developers as opposed to just throwing code over the fence. Another decision that worked well for them was to use existing standards when appropriate. For example, by implementing POSIX.1e ACLs, the Samba server was able to use them with only minor modification. In the future they would like to increase performance and improve the Extended Attribute implementation.

More information can be found at *http://www.trustedbsd.org.*

Integrating Flexible Support for Security Policies into the Linux Operating System

Stephen Smalley, NAI Labs

MACs are able to solve many of the security limitations in current systems but have not yet become widely used. Part of this can be attributed to a lack of flexibility in current systems. Working with Secure Computing Corporation, the NSA developed a flexible architecture called Flask. The security logic in Flask is cleanly separated from the enforcement mechanisms, so users can develop policies based on their particular requirements. The NSA contracted with NAI Labs to create a sample security policy packaged with the software. This sample implementation combines type enforcement, role-based access control (RBAC), and multi-level security (MLS). In SELinux, every operation can have a unique security policy. This allows extremely fine-grained access controls tailored for different uses. Each time an operation is performed the access is revalidated, which means that policy changes take effect immediately.

In benchmarks SELinux showed large performance penalties on some operations, but overall the effect was negligible. Kernel compilation showed a 4% increase in system time and no significant increase in wall time. The benchmarks we done using the very extensive default policy.

More information can be found at *http://www.nsa.gov/selinux*.

SCRIPTING

Summarized by Brandon Ching

A Practical Scripting Environment for Mobile Devices

Brian Ward, Department of Computer Science, University of Chicago

Ward began his presentation by labeling some of the major problems associated with programming for handheld applications. The amount of computational activity and resources available in handheld devices is severely limited because of their size. And their small screen size also presents the problem of proper graphical display. Mobile devices usually do not do any real computation; rather, they primarily display information, which makes proper graphical representation so important.

In lieu of directly tackling such dilemmas, Ward has come up with a parser/compiler similar to PHP, which is called HHL and a virtual machine interpreter called VL. Economizing on space and eliminating redundancy optimizes HHL. It is coded in ANSI Standard C and works like any UNIX compiler.

With these tools, Ward hopes scripting for mobile handheld devices will become more efficient and productive.

Nickle: Language Principles and Pragmatics

Bart Massey, Computer Science Department, Portland State University; Keith Packard, SuSE Inc.

Bart Massey unveiled his and Keith Packard's new C-like programming language called Nickle, revealing the purpose and features of this numerical applications program.

The three main purposes for the Nickle language are calculation, experimentation (algorithms), and prototyping. Massey said that new programming languages should exhibit four basic characteristics. They should serve a useful purpose, serve people other than their creators, be the best language for the given task, and draw on the best practices. According to Massey, Nickle does all of these. With features such as interactive byte code, "C" style programming, powerful numeric types, useful language extensions, and user level threads, Nickle can serve a variety of uses.

USER SPACE

Summarized by Rosemarie Nickles

User-Level Checkpointing for LinuxThreads Programs

William R. Dieter and James E. Lumpp, Jr., University of Kentucky

Dieter introduced the first system to provide checkpointing support for multi-threaded programs that use LinuxThreads, the POSIX-based threads library for Linux. Checkpointing saves the state of a process so that in the event of a system hardware or software failure all would not be lost.

Implementation of the multi-threaded checkpointing library:

- To take a checkpoint all threads are blocked except the main thread. This thread saves the process state and unblocks all the remaining threads.
- To recover from a checkpoint, the checkpointing library restarts the threads which were running when the checkpoint was taken. These threads are blocked until the main thread has loaded the process state from the checkpoint, at which time the threads continue to run from the checkpoint.

The checkpoint library adds little overhead except when taking a checkpoint. This overhead is in proportion to the size of the address space. It is also easy to use. A C programmer needs only to add two lines of code. Source code is available from *http://www.dcs.uky.edu/~chkpt* and *http://mtckpt.sourceforge.net.*

Building an Open Source Solaris-Compatible Threads Library

John Wood, Compaq Computer (UK) Ltd.

This presentation compared Solaris threads to POSIX threads. John Wood discussed the unique Solaris functionality and how to implement:

- Daemon threads
- Join any thread
- Thread suspend and continue

Solving the problem by building an open sourced Solaris compatible threads library would:

- Enable applications that use the Solaris threads API to be ported
- Be an alternative to reworking applications to use POSIX threads
- Not solve generic porting issues

Building an open source Solaris-compatible threads library would eliminate the expense of rewriting the generally non-portable applications that use the Solaris threads application-programming interface.

Questions? Write to *Solaris.complib@compaq.com* or visit *http://www.opensource.compaq.com* or *http://www.tru64unix.compaq.com/complibs/documentation/html/TITLE.HTM*.

Are Mallocs Free of Fragmentation?

Aniruddha Bohra, Rutgers University; Eran Gabber, Lucent Technologies, Bell Labs

During a comparison study the conclusion was made that mallocs are not free of fragmentation. Nine mallocs were tested with both Hummingbird and GNU Emacs, and the fragmentation varied but none was fragmentation free. PHK/BSD malloc version 42 took first place with a 30.5% fragmentation rate during the Hummingbird test. Doug Lea's malloc version 2.6.6 took the top honors in the GNU Emacs test with a fragmentation rate of 2.69%. PHK/BSD fell to a close fifth with a fragmentation rate of 3.65% in the GNU Emacs test. The worst malloc remained consistent in both tests. Sun OS version 5.8 caused a fragmentation rate of 101.48% in GNU Emacs and failed to finish in the Hummingbird test after causing a heap overflow. Developers should be aware that mallocs are not created equal and should pick one that works well for their workload.

This presentation ended with a plea for further research to understand why certain malloc implementations cause excessive fragmentation.

The Hummingbird and Emacs memory activity traces, the source for the driver program, and the modified bin buddy allocator are available at *http://www.bell-labs.com/~eran/malloc/*.

USER ENVIRONMENT

Summarized by William R. Dieter

Sandboxing Applications

Vassilis Prevelakis, University of Pennsylvania; Diomidis Spinellis, Athens University

Sandboxing helps improve security when running large applications on untrusted data by limiting what resources the program can access. Determining which kinds of access should and should not be allowed is difficult, however. The File Monitoring and Access Control (FMAC) tool helps build sandboxes for applications. To build a sandbox, the user runs the application under FMAC in passive mode on some known safe input. FMAC records the files requested and passes them through to the system. FMAC constructs an access control list (ACL) based on the recorded file requests and uses it to generate a sandbox specification.

When FMAC is run with the sandbox specification it uses chroot to limit access to a particular directory then mounts a special NFS file system on that directory. The special NFS file server allows programs run in the sandbox to access files based on the sandbox specification. Users can view the automatically generated ACL files to see which resources a program uses or modify them by hand to generalize or further limit what resources the program can access. The FMAC tool is designed to balance risk with cost. It is portable, easy to configure, and provides an "adequate level of security" for many users.

One audience member was interested in "session" sandboxes for applications that share multiple files. Vassilis Prevelakis replied that session sandboxes are unnecessary because the sandbox mechanism just filters the view of the file system. There is no need to make multiple copies of files. Other questions related to how well the learning phase covers what the application will try to use once it is in the sandbox. Vassilis said that can be a problem. For example, in one case a user did not access the Netscape help files during the learning session. They were blocked when Netscape ran in the sandbox. Dynamically asking the user if an action should be allowed is generally not safe, because users are confronted with so many pop-up windows they often automatically click "OK."

Building a Secure Web Browser

Sotiris Ioannidis, University of Pennsylvania; Steven M. Bellovin, AT&T Labs – Research

Due to the explosion in the exchange of information, many modern security threats are data driven. Mail viruses and macro viruses hide inside documents that users want to see and then execute with the same permissions as the users. Sotiris Ioannidis described how SubOS, which is based on OpenBSD, provides a finer-grained level of control than the standard UNIX permission model. When an object, typically a file, arrives at the system, SubOS assigns it a sub-user ID, which corresponds to an access control list. Any time the object is copied, the copies inherit the sub-user ID. Any program that touches the file is limited to the permissions allowed by the sub-user ID.

Sotiris also described a secure Web browser built on SubOS. The browser assigns sub-user IDs to all the objects it downloads. Objects that have valid certificates from trusted sources are given more permissions than untrusted objects. If an object contains code, like JavaScript or Java, the object is run in a separate process with the permissions allowed by its sub-user ID. The damage downloaded scripts can do is limited by their sub-user ID.

When asked what happens if a process accesses two files with different sub-user IDs or communicates through a pipe, Sotiris responded that the process would get the intersection of the two processes' permissions. He also said that, although pipes are currently not covered, all transfer of data should be authenticated. It was pointed out that the application receiving the data from the network needs to help assign permissions to incoming information. Sotiris replied that although the application needs to help with the assignment, once the assignment is made the operating system takes over.

Citrus Project: True Multilingual Support for BSD Operating Systems

Jun-ichiro Hagino, Internet Initiative Japan Inc.

Jun-ichiro Hagino explained how Citrus adds new libraries to NetBSD, and soon OpenBSD, to help applications support multilingual character sets. Character set encodings have evolved from the original 7-bit ASCII to 8-bit encodings that handle most European languages, and to multibyte character sets for larger character sets. External character sets are used outside the program when characters are stored in files. Internal character sets represent characters in memory. Both internal and external character sets continue to evolve, and it is difficult to predict what future character set encodings will look like.

To improve compatibility with existing and future character sets Citrus does not impose a particular internal or external character set encoding. Instead, it dynamically loads a module at runtime to handle a particular user's locale settings. Not imposing a character set helps avoid losing information. For example, multilingual support libraries that use Unicode internally can lose information because some Asian characters that represent different words in different Asian languages map to the same Unicode code points. Citrus avoids this problem by only converting between formats when explicitly requested.

One audience member asked how to separate Citrus from NetBSD to port it to other platforms. Jun-ichiro Itojun said the CVS tree is available and a Citrus Web page will be updated with more information. Another audience member asked how application integration is progressing. Under X11 with KDE and GNOME the window managers can handle it. The older UNIX utilities like vi still do not have multilingual support.

KERNEL

Summarized by Kenneth G. Yocum

Kqueue: A Generic and Scalable Event Notification Facility

Jonathan Lemon, FreeBSD Project

Applications typically receive notifications of events, such as I/O completions, through a select call or by polling. It has been shown that with thousands of event sources, e.g., a Web server with thousands of connections, selective calling/polling does not scale. Kqueue provides a new API for event notification. Kqueue also allows the application to specify filters, so that atypical events can be delivered to the application, including AIO and signals. It was designed to be cheap and fast. Though the setup cost is higher than for polling, it is cheaper when dealing with many possible events. Kqueue filters can also be used to deliver device events or periodic timers.

Improving the FreeBSD SMP Implementation

Greg Lehey, IBM LTC Ozlabs

SMP support in FreeBSD has been woefully inadequate. Typically, only one process could be in the kernel at a time, and blocked interrupts across all processors. Essentially SMP support was provided by one Big Lock. This paper describes their work in applying fine-grain locking for better SMP performance. One problem: interrupt handlers can't block, because they don't have a process context. So give them one, and call it the interrupt thread. Current work is underway to migrate current interrupt handlers to use mutex's in place of calls to spl<whatever>. Though too early for performance numbers, expect the system to scale beyond 32 processors.

Page Replacement in Linux 2.4 Memory Management

Rik van Riel, Conectiva Inc.

The Linux 2.2 virtual memory (VM) subsystem has interesting performance limitations in some circumstances. For instance, pages will be reclaimed from the file cache but not from a day-old idle process. Because page-age information is only accumulated during low-memory situations, a spike in VM activity can cause the system to throw out recently accessed pages. With 2.4 they want to support fine-grain SMP and machines with more than 1GB of memory. They unified the buffer cache, and reintroduced page aging. In general the results have been well received. Performance and stability seem to have improved, though no figures are reported.

STORAGE

Summarized by Adam Hupp

User-Level Extensibility in the Mona File System

Paul W. Schermerhorn, Robert J. Minerick, Peter Rijks, and Vincent W. Freeh, Department of Computer Science and Engineering, University of Notre Dame

The Modify-on-Access (Mona) file system is a new model for manipulating data streams. Transformations are defined on the input and output streams of a file. This allows the system to transparently modify file streams during reads and writes. They are implemented as user-mode shared libraries, as well as at the kernel level. For example, a PHP transformation can automatically parse a PHP template and output the resulting

HTML. An FTP transformation could allow users to manipulate remote files as easily as local ones. There are significant advantages for programmers as well. Since common operations can be transparently layered upon each other, developers will be able to use complex functionality through normal I/O mechanisms. It is unnecessary to learn new APIs to use existing components, and writing new components is very simple.

Mona had little overhead when compared to a standard ext2 file system. When tested with complex operations Mona quickly begins to outperform UNIX pipes at equivalent tasks. This speedup (up to 65%) is due to stacked transformations sharing address space and eliminating task switch and buffer copying overheads.

More information is available at *http://www.cse.nd.edu/~ssr/projects/mona.*

Volume Managers in Linux

David Teigland, Heinz Mauelshagen, Sistina Software, Inc.

Volume managers allow disks to be organized logically instead of as fixed sizes. They are becoming critical in expanding Linux systems to the enterprise. This paper gives an overview of volume management software in Linux and some of the developments that are currently being worked on.

The LVM (Logical Volume Manager) and MD (Multi-Disk) driver are the primary volume managers used in Linux. They allow RAID and logical administration of disks, which increases performance and reliability. The latter is useful in both small and large systems. When there is limited disk space, such as on a laptop, the LVM can be used to reallocate partitions that are wasting space. In large systems, the ability to replace disks without bringing down the system is extremely useful.

There are new features being developed for the volume management systems under Linux. The ability to take snapshots of the file system at a point in time was recently added to the LVM. This enables backups to be taken without the risk that a file will be written to. It does not solve the problem of applications leaving data in an inconsistent state, but is a good step in easing backup difficulties. Another new feature currently being implemented is metadata export. This tells the system about the underlying disks and can be used to intelligently place data for optimal performance. In clusters, work is being done on sharing volume management across all nodes. This allows the volumes to be modified on any node, and changes will be consistently propagated across all other nodes.

For more information, see *http://www.sinista.com.*

The Design and Implementation of a Transparent Cryptographic File System for UNIX

Giuseppe Cattaneo, Luigi Catuogno, Aniello Del Sorbo, and Pino Persiano, Dipartimento di Informatica ed Appl., Università di Salerno

Current implementations of distributed file systems lack good protections against eavesdropping and spoofing. The Transparent Cryptographic File System (TCFS) has been developed to provide strong security while remaining simple to use. Files are stored in encrypted form so the remote server will not have access to their contents. Any unauthorized attempt at modification will be immediately noticed. It is implemented in Linux as a layer on top of the VFS, and uses the NFS protocol to communicate with the server.

Key management is often difficult in cryptographic systems, and TCFS has a variety of ways to deal with this. It supports raw keys, basic keys, shared keys, and Kerberized keys. Basic keys use the person's login password as they key. The Kerberized keys allow a user to obtain a ticket from a TCFS key server which then provides access to the files. Shared keys used a secret splitting algorithm in which *n* shares are needed to recreate the original key. Users provide their shares to the kernel, and when enough shares have been received, the file is retrieved. In the future they would like to improve the performance to be closer to standard NFS.

GRAPHICS

Summarized by Rosemarie Nickles

Design and Implementation of the X Rendering Extension

Keith Packard, Xfree86 Core Team, SuSe Inc.

The Xfree86 Core Team picked up the challenge laid down at the 2000 USENIX Technical Conference, where a presentation outlined the state of the X rendering environment and the capabilities necessary to bring X into the modern world. Xfree86 brought forth the X Rendering Extension (Render) with some help with the final architecture from KDE, Qt, Gdk, GNOME and OpenGL. Render replaces the pixel-value-based model of the core X [SG92] rendering system with a RGB model. The examples shown were crisp and clear.

Topics discussed were:

- Anti-Aliasing and Image Compositing
- Rendering Model - Operator
- Premultiplied Alpha
- Basic Compositing in Render
- Text Rendering
- Client-Side Glyphs
- Xft Library
- Polygons, Trapezoids, and Smooth Polygons
- Image Transformation

SCWM: AN EXTENSIBLE CONSTRAINT-ENABLED WINDOW MANAGER

Greg J. Badros, InfoSpace, Inc.; Jeffrey Nichols, School of Computer Science, HCI Institute, Carnegie Mellon University; and Alan Borning, University of Washington

Scwm – the Scheme Constraints Window Manager – pronounced "swim," is a complete window manager built for X/11. Scwm's most notable feature is constraint-based layout. Scwm not only embeds the Cassowary Constraint Solving Toolkit for layout constraints but also has a graphical user interface which employs an object-oriented design. Users can create constraint objects or create new constraint classes from existing classes.

Some of the existing constraints were demonstrated and a few of them follow:

- Vertical alignment: This aligns the left edge of one window with the right edge of another
- Constant height/width: If one window were resized the window/windows constrained with it would resize also
- Horizontal/Vertical separation: No matter where a window was moved the window/windows joined in the constraint would always be to the left or above it

Constraints can be enabled or disabled by using the constraint investigation window. Checkboxes are used to enable/disable constraints, and a delete button removes the constraint. There was no question which constraint was being targeted: each time the mouse passed over a constraint in the investigator, the windows related by the constraint were highlighted by a brightly colored line around them.

Scwm can be downloaded from *http://scwm.sourceforge.net.*

THE X RESIZE AND EXTENSION – RANDR

Jim Gettys, Cambridge Research Laboratory, Compaq Computer Corporation; Keith Packard, Xfree86 Core Team, SuSe Inc.

Have you ever tried to look at your desktop monitor sideways or upside down? The solution is the Resize and Rotate extension (RandR). RandR is designed to allow clients to modify the size, accelerated visuals, and the rotation of an X screen. Laptops and handheld devices need to change their screen size when hooked up to external monitors with different resolutions. RandR allows these changes with simple modifications. A prototype of the RandR is functioning in the TinyX X implementation. This presentation was given using a Compaq iPAQ H3650 Handheld Computer with a HP VGA out PCMCIA card, using Familiar Linux .4, Xfree86 4.1 TinyX Server with RandR extension and MagicPoint Presentation tool.

RESOURCE MANAGEMENT

Summarized by Rosemarie Nickles

PREDICTABLE MANAGEMENT OF SYSTEM RESOURCES FOR LINUX

Mansoor Alicherry, Bell Labs; K. Gopinath, Department of Computer Science & Automation, Indian Institute of Science, Bangalore

Alicherry described the Linux scheduler to his audience. He detailed the three scheduling policies: the static priority, the preemptive scheduling, and the "counter" for SCHED_Other. He then turned his attention to resource containers, describing how they are accessed and their parent-to-child hierarchy, as well as the scheduler framework. He explained the source code for changing the CPU share of a container and shell using resource container. The performance overhead chart had the time taken for various operations broken down into microseconds.

For more information: *mansoor@research.bell-labs.com.*

SCALABLE LINUX SCHEDULING

Stephen Molloy and Peter Honeyman, CITI, University of Michigan

Linux uses a one-to-one thread model, which is implemented easily but potentially overloads the kernel's default scheduler. Servers run multi-thread applications. Experiments conducted by IBM showed that a scheduler for heavily threaded workloads could dominate system time. The Linux scheduler design assumes a small number of ready tasks, such as:

- Simple design means easy implementation
- On runtime
- It performs many of the same calculations on each invocation

The goals are:

- To make the scheduler fast for both large numbers of tasks (server environment) and small numbers of tasks (desktop environment)
- To provide an incremental change to keep current criteria, maintain current interfaces, and preserve goodness metric
- To understand reasons for any performance differences

The solution is to use a sorted table instead of an unsorted queue and only to examine tasks in the highest populated list, falling through to the next list if strictly necessary.

Any questions?

Steve Molloy: *smolloy@umich.edu*
Peter Honeyman: *honey@citi.umich.edu*
CITI: *http://www.citi.umich.edu*
IBM Linux Tech Center: *http://www.linux.ibm.com*

A UNIVERSAL DYNAMIC TRACE FOR LINUX AND OTHER OPERATING SYSTEMS

Richard Moore, IBM, Linux Technology Center

A universal dynamic trace can operate in kernel or user space. It will work under the most extreme conditions at interrupt time or task time or even between contests, and it operates in an MP environment. It also can be used to trace code or data usage. It is dynamic because of the ability to insert tracepoints at runtime without the need to modify or prepare traced code in advance. Actions taken when the tracepoint fires are customizable at runtime. Thus there's a need for a debugging engine to be interfacing with a standard system tracing mechanism. Dynamic Probes provides the debugging engine.

For more information:

Mailing list: *dprobes@oss.software.ibm.com*

Web page: *http://oos.software.ibm.com/developerworks/opensource/linux/projects/dprobes*

General Track

Operating Systems

Summarized by Kartik Gopalan

Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor

Jeremy Sugerman, Ganesh Venkitachalam, Beng-Hong Lim, VMware Inc.

This paper won the Best Paper Award in the General Track. In a very well presented talk, Jeremy Sugerman described VMware Workstation's approach to virtualizing I/O devices and explained various optimizations that can improve the throughput of the virtualized network interface.

The talk began with a recounting of the concept of virtual machines pioneered by IBM in its mainframe machines. The concept of virtual machines is still beneficial in the context of modern-day desktop PCs due to users' need to run applications from multiple operating systems simultaneously. It can also be useful for server consolidation by enterprises and services providers in order to better utilize resources and ease system manageability.

Jeremy described the VM architecture and the mechanism used for virtualizing the network interface. The virtual NIC appears as a PCI-Ethernet controller to the guest OS and can either be bridged to a physical network connected to the host or connected to a virtual network created within the host. All packets sent out by guest OS are directed to the VMApp by a VMDriver, which transmits the packet out on the physical network through a VMNet driver.

The TCP throughput provided by such an approach on a 733MHz Pentium machine was only 60Mbps as compared to native throughput of 90Mbps. The main reason for lower throughput by VM was due to I/O space accesses requiring world switch to VMApp and the time spent processing within the VMApp. The key strategy to improve performance is to reduce the number of world switches. With optimizations such as making VMM directly handle I/O, not requiring host hardware, clustering packets before sending, and using shared memory between VMNet driver and VMApp, the TCP connection is able to saturate the network link.

This work essentially showed that VMware Workstation's achievable I/O performance strikes a good balance between performance and modern-day desktop compatibility. For more information, visit *http://www.vmware.com.*

Magazines and Vmem: Extending the Slab Allocator to Many CPUs and Arbitrary Resources

Jeff Bonwick, Sun Microsystems; Jonathan Adams, California Institute of Technology

Jeff Bonwick first reviewed the current state of slab allocation mechanisms. Slab allocators perform object caching to reuse states of previously created objects. However, there are two disadvantages: global locking doesn't scale to many CPUs and allocators cannot manage resource other than kernel memory.

To address scalability, Jeff proposed the "magazine layer" which consists of magazines and depots. Magazines are basically per-CPU caches whereas depots keep a global stockpile of magazines. Each CPU's allocations can be satisfied by its magazine until the magazine becomes empty, at which point a new magazine is reloaded from the depot. The performance measurements indicated almost perfect scaling properties of magazine layer with increasing number of CPUs.

Virtual address allocation is just one example of a more general resource allocation problem, where resource is anything that can be described by a set of integers. Jeff proposed a new general-purpose resource allocator called "Vmem," which provides guaranteed constant-time performance with low fragmentation and linear scalability. Vmem eliminates the need for special purpose allocators, such as a process ID allocator, in the operating system. The implementation details and performance results were presented. Vmem was shown to provide constant-time performance even with increasing fragmentation.

Following this, Jonathan Adams presented a user-level memory allocation library called "libumem." Libumem is a user-level port of the kernel implementation of magazine, slab, and Vmem technologies. Some of the porting issues reported dealt with replacing CPU ID with thread ID, handling memory pressure, supporting malloc(3C) and free(3C), and needing lazy creation of standard caches. Libumem was shown to give superior malloc/free throughput performance compared to hoard, fixed, and original mtmalloc, ptmalloc, and libc.

Magazines and Vmem are part of Solaris 8. The sources are available for free download at *http://www.sun.com.*

Measuring Thin-Client Performance Using Slow-Motion Benchmarking

S. Jae Yang, Jason Nieh, and Naomi Novik, Columbia University

In this talk, Naomi Novik presented the technique of slow-motion benchmarking for measuring the performance of thin-client machines. First, she introduced the concept of thin clients. Thin clients are designed to provide the same graphical interfaces and applications available on traditional desktop computers while centralizing computing work on powerful servers. All application logic is executed on the server, not on the client. The user interacts with a lightweight client that is generally responsible only for handling user input and output, such as receiving screen display updates and sending user input back to the server over a network connection.

The growing popularity of thin-client systems makes it important to develop techniques for analyzing their performance. Standard benchmarks for desktop system performance cannot be used to benchmark thin-client performance since applications running in a thin-client system are executed on the server. Hence these benchmarks effectively only measure the server's performance and do not accurately represent the user's experience at the client-side of the system. To address this problem, Naomi presented slow-motion benchmarking, a new measurement technique for evaluating thin-client systems.

The performance of a thin-client system should be judged by what the user experiences on the client. Direct instrumentation of thin clients is difficult since many thin-client systems are quite complex. In slow-motion benchmarking, performance is measured by capturing network packet traces between a thin client and its respective server during the execution of a slow-motion version of a standard application benchmark. Slow-motion execution involves altering the benchmark application at the server end. This is done by introducing delays between the separate visual components of that benchmark so that the display update for each component is fully completed on the client before the server begins processing the next one. These results can then be used either independently or in conjunction with standard benchmark results to yield an accurate and objective measure of user-perceived performance for applications running over thin-client systems.

Naomi concluded the talk by presenting slow-motion benchmarking measurements of Web server and video playback benchmarks on client/server systems that included a Sun Ray thin client machine and a Sun server.

STORAGE I

Summarized by Joseph Spadavecchia

The Multi-Queue Replacement Algorithm for Second-Level Buffer Caches

Yuanyuan Zhou and James Philbin, NEC Research Institute; Kai Li, Princeton University

Yuanyuan Zhou presented the Multi-Queue Replacement Algorithm (MQ) for second-level buffer caches. Almost all second-level buffer caches use locality-based caching algorithms, such as LRU. These algorithms do not perform well as second level buffer caches, because they have different access patterns than first-level buffer caches – accesses in the second level are missing from the first. Almost all of today's distributed multi-tier computing environments depend on servers that usually improve their performance by using a large buffer to cache data. There is a strong need for a better second-level buffer cache replacement algorithm.

The authors' research shows that a good second level buffer cache replacement algorithm should have the following three properties: minimal lifetime, frequency-based priority, and temporal frequency.

The minimal lifetime constraint means that warm blocks stay in the buffer cache at least a certain amount of time for a given workload. Frequency-based priority, as its name suggests, assigns blocks priority based on their access frequency. Temporal frequency is used to remove blocks that are not warm.

The MQ algorithm satisfies the three properties listed above and has O(1) time complexity. MQ uses multiple LRU queues to maintain blocks with different access frequencies for different periods of time in the second-level buffer cache. MQ is also simpler to implement than FBR, LRFU, and LRU-K.

The authors did trace-driven simulations to show that MQ outperforms LRU, MRU, LFU, FBR, LRU-2, LRFU and 2Q as a second-level buffer cache replacement algorithm, and that it is effective for different workloads and cache sizes. In some cases MQ yields a 53% improvement over LRU and a 10% higher hit ratio than FBR.

The proof is in the implementation. The authors tested the performance by implementing MQ and LRU on a storage server with Oracle 8i Enterprise Server as the client. The results obtained using TPC-C benchmark on a 100GB database show that MQ improves the transaction rate by 8–11% over LRU. For LRU to achieve the same performance as MQ requires that the server's cache size be doubled.

Design and Implementation of a Predictive File Prefetching Algorithm

Thomas M. Kroeger, Nokia Clustered IP Solutions; Darrell D. E. Long, University of California, Santa Cruz

Kroeger discussed the design and implementation of a predictive file prefetching algorithm. Research has shown that the patterns in which files are accessed can predict upcoming file accesses. Almost all modern caches do not take file access patterns into account. Heuristics that expect sequential accesses cannot be applied to files, because the concept of a file does not have a successor. Hence, modern caches fail to make use of valuable information that can be used to reduce I/O latency.

Previously the authors developed a compression-modeling technique called Partitioned Context Modeling (PCM) that monitors file access to predict upcoming requests. PCM works in a linear state space through compression, but experimentation showed that it does not predict far enough into the future. Therefore, the authors developed Extended Partition Context Modeling (EPCM) that predicts much farther into the future.

The authors implemented predictive prefetching systems (PCM and EPCM) in Linux and tested them with the following four application-based benchmarks:

1. Andrew Benchmark
2. GNU ld of the Linux kernel
3. Glimpse index of /usr/doc
4. Building SSH

The Andrew Benchmark is a small build. Though dated, it is widely used and accurately portrays the predictive relationship between files. The GNU ld of the Linux kernel was used to represent a workload of non-sequential file accesses. The Glimpse indexing of /usr/doc generated a workload representing a traversal of all files under a given directory. Finally, the building of SSH 1.2.18 through 1.2.31 was used to represent the compile edit cycle. The system was able to train on the initial version (1.2.18) and then used that training on sequentially-modified versions (1.2.19 – 1.2.31).

The results of testing show that I/O latency reduced by 31–90% and elapsed time reduced by 11–16%. With EPCM, the Andrew Benchmark, GNU ld, Glimpse, and SSH saw elapsed time improvements of 12%, 11%, 16%, and 11%, respectively. I/O latency was improved as much as 90%, 34%, 31%, and 84%, respectively.

Question: Were any tests done on a multi-user system where accesses are not associated with the tasks of a single user? It seems that it would be harder to predict file access with multiple users. Answer: Tests like these have not been done yet.

Extending Heterogeneity to RAID Level 5

T. Cortes and J. Laborta, Universitat Politécnica de Catalunya

Cortes described work on extending heterogeneity to RAID level 5 (RAID5), which is one of the most widely used types of disk arrays. Unfortunately, there are some limitations on the usage of RAID5. All disks in a RAID5 array must be homogeneous. In many environments, especially low-cost ones, it is unrealistic to assume that all disks available are identical. Furthermore, over time disks are upgraded and replaced resulting in a heterogeneous unit. According to studies by IBM, disk capacity nearly doubles while prices per MB decrease by 40% every year. Consequently, it is neither convenient nor efficient to maintain a homogeneous RAID5 disk array.

There are some projects that have already focused on solving this problem; however, they deal only with multimedia systems. The solution the authors described is intended for general purpose and scientific settings, though it also works well for multimedia applications.

The authors presented a block distribution algorithm called AdaptRaid5 that they used to build disk arrays from a set of heterogeneous disks. Surprisingly, in addition to providing heterogeneity, AdaptRaid5 is capable of servicing many more disk requests per second than RAID5. This is because RAID5 assumes that all disks have the lowest common speed, whereas AdaptRaid5 does not.

Experimental results were shown comparing the performance of traditional RAID5, RAID5 using only fast disks (OnlyFast), and AdaptRaid5. AdaptRaid5 significantly outperformed RAID5 and OnlyFast for capacity evaluation, full-write, and small-write performance measures. However, with more than six disks OnlyFast performed better than AdaptRaid5 for read, and real-workload performance measures. This is because AdaptRaid5 must account for slow disks, whereas OnlyFast cannot.

The authors measured real-workload performance by using a trace file supplied by HP. Performance gains obtained using AdaptRaid5 versus RAID5 over five disks were almost 30% for reads, and 35% for writes. AdaptRaid5 versus OnlyFast performance gains ranged from nearly 30% for reads to 39% for writes. OnlyFast had an approximate 3% read performance gain over AdaptRaid5 when eight fast disks were used, but this is because slow disks are never used in OnlyFast.

TOOLS

Summarized by Peter da Silva

Reverse-Engineering Instruction Encodings

Wilson Hsieh and Godmar Beck, University of Utah; Dawson R. Engler, Stanford University

Hsieh presented this paper. The problem the authors were trying to solve was how to efficiently produce code generators for just-in-time (JIT) compilers like Kaffe. The JIT compiler has to produce efficient instruction sequences quickly and reliably; creating the tables for the code generator from an instruction sheet is complex and error-prone.

Most systems, however, already have a program that knows about the instruction set of the computer, the assembler. By generating instruction sequences and passing them through the assembler, their system, DERIVE, can produce tables that describe the instruction set and can be used to drive code generators.

Wilson described how DERIVE takes a description of the assembly language and repeatedly generates instruction sequences that step-by-step probe the underlying instruction set to derive register fields, opcode fields, and labels. There are three phases: the register solver, immediate solver, and jump solver.

The register solver tests each argument at a time, sequencing through all possible registers. In a RISC CPU this is simple, and a single pass through the assembler can provide all possible bitmaps for analysis. But for a complex instruction set like the Intel x86 many combinations of registers have unique encodings or are even illegal, so the assembler has to be called over and over again for each combination.

The immediate solver and jump solver work similarly, calling the register solver to extract the encodings. The immediate solver (which also handles absolute jumps) performs a linear search through the possible arguments to find the maximum size, then solves each possible argument size separately. The jump solver does a similar job, except it has to generate appropriate labels and adjust for scaling.

Solving an instruction set can take between 2.5 minutes and four hours (for the Intel x86 architecture), depending on the complexity of the encoding.

The generated tables are a set of C-like structures that are passed to a code generator, which produces C macros to generate the final code. These tables are surprisingly efficient: they were able to reduce the size of the Kaffe code generator for the x86 architecture by 40% with one day's work.

Source code is available from *http://www.cs.utah.edu/~wilson/derive.tar.gz.*

An Embedded Error Recovery and Debugging Mechanism for Scripting Language Extensions

David M. Beazley, University of Chicago

What happens if you have an error in C or C++ code called from a scripting language? Well, normally if you have an error in a high-level language, the interpreter gives you a nicely formatted backtrace that shows exactly where your program died. Similarly, if your C code crashes you get a core dump that can be examined by a debugger to produce a nicely formatted backtrace showing you exactly where your program died.

In a scripting language extension, you get a low-level backtrace of the high-level-language stack, which generally consists of layer after indistinguishable layer of the same three or four interpreter routines over and over again. Digging useful information out of this can be challenging.

To solve this problem, WAD (Wrapped Application Debugger) runs as a language extension itself and sets up signal handlers for all the common traps, such as SIGSEGV, SIGBUS, and so on. When an exception occurs, WAD unrolls the stack and generates a formatted dump of the low-level code, using whatever debugging information is available to it in symbol and debugging tables, then simulates an error return to the highest level interpreter stack frame it can find and passes this dump back as the error text.

At this point the scripting language itself can unroll its own stack the rest of the way and pass the combined set of stack traces to the programmer.

No relinking and no separate debugger are necessary.

Beazley proceeded to demonstrate the debugger for both Tcl and Python. For the first, a small wish program opened a Tk window that provided radio buttons to select exactly what kind of exception to use, using an extension in C that simply produced the requested exception and let WAD and the Tk error handler pop up a window containing the combined stack trace. For the second, he used a Web server running Python extensions, with an error handler that dumped the stack trace to the browser and the Web server's error log.

There are problems to be worked out. Since the debugger doesn't have the same intimate knowledge of the code as the interpreter's error handler, it can leak memory, lose locks, lose open files, and so on. Still, my biggest disappointment is that it's not available for Tru64/Alpha but only for Solaris/Sparc and Linux/x86. For more information, visit *http://systems.cs.uchicago.edu/wad.*

Interactive Simultaneous Editing of Multiple Text Regions

Robert C Miller and Brad A. Myers, Carnegie Mellon University

Miller started out by thanking USENIX for supporting his work.

Then he described the problem he was trying to solve: repetitive text editing is error-prone, even with the assistance of macros, regular expressions, and language-sensitive editors. This paper described a tool that attacks the problem from a different direction, using an interactive program that provides the user with immediate feedback while performing the same editing operation in multiple places in a file.

Lapis is a simultaneous editor; the user selects parts of a region and edits it, and the same operation is performed simultaneously in the same place in all similar regions.

Editing is the easy part. Identifying the fields to be edited is harder. Lapis solves this problem by providing instant feedback and examples. The user identifies a section of the region by selecting it, and then Lapis generalizes the selection and highlights it in all fields. If the program guesses wrong – too much or too little selected in some field – the user can provide more samples to home in on the desired selection.

Splitting the file up into regions can be handled similarly, by selecting a number of examples and entering simultaneous editing mode on these regions. Alternatively a pattern can be selected directly from a nested list in the lower right of the Lapis window, and it will automatically select all the matching regions.

Performance is a problem: the heuristics Lapis uses to recognize patterns and fields are expensive. Lapis solves this by finding all interesting features of each region when the file is split up, then adding new features as the user continues to edit them. Features are never removed from this list; it's faster to skip over false positives.

Robert then described a series of experiments at CMU, where undergraduates were given a group of simple editing jobs and asked to solve them using simultaneous editing and more traditional tools. To stack the deck against himself, he had the students perform the operation using Lapis first, so they were already familiar with the problem when they switched to their traditional tools.

For as few as 3–10 records Lapis was already faster than traditional tools, for users who had never used Lapis before.

A Java implementation is available from *http://www.cs.cmu.edu/~rcm/lapis.*

WEB SERVERS

Summarized by Kenneth G. Yocum

High-Performance Memory-Based Web Servers: Kernel and User-Space Performance

Richard Neves, Philippe Joubert, ReefEdge Inc.; Robert King, John Tracey, IBM Research; Mark Russinovich, Winternals Software

This is a four-year-old IBM effort to improve Web performance. The first kernel-mode Web server was produced in 1998. It has made its way into S/390, AIX, Linux, and Windows. The goals were to identify the performance gap between user mode and kernel mode on multiple production OSes without modifying the kernel (TCP/IP stack, drivers, or hardware). They identified three first order performance issues with user-mode servers: data copies, event notification, and data paths. User-mode approaches include reducing memory copies and performing checksum offloading. This means using mechanisms like Fbufs, IO/Lite, a transmit file primitive, and Kqueue.

IBM introduced AFPA (Adaptive Fast Path Architecture), the kernel-mode engine, in 1997. It is integrated with TCP/IP stack and the file system. Other systems included TUX, early Linux kernel-mode Web servers, the Lava hit-server, and Cheetah in the Exokernel work. In AFPA the HTTP module is separated for portability, and it can still run with user-mode Apache or proxy cache. AFPA supports multiple protocols, not just HTTP. Kernel manages zero-copy cache.

The test platform was 12 two-way 450MHz Xeon clients and a uniprocessor server. It achieved 1.2Gbps performance. In summary, user mode is about 3.5 times slower than kernel mode. Interrupt-based architectures are 12% faster than thread-based ones. Zero-copy doesn't help much with requests less than 4K, but direct integration with the TCP/IP stack can improve performance by 55%.

Kernel Mechanisms for Service Differentiation in Overloaded Web Servers

Thiemo Voigt, Swedish Institute of Computer Science; Renu Tewari, Douglas Freimuth, IBM T.J. Watson Research Center; Ashish Mehra, iScale Networks

The Internet is quickly growing and requiring support for new services that depend on highly available servers. Server overload is a problem, and people won't pay for its solution. Traditional servers provide marginal overload protection, but that's not good enough. To get predictability they provide three mechanisms: TCP SYN policing, prioritized listen queue, and URL-based connection control. As with most systems papers, there are three design principles: don't waste cycles, minimize changes to network subsystem, and be able to implement these mechanisms on servers and on Layer4/7 intermediary switches.

The three mechanisms provide support at increasing levels of consumed resources. TCP SYN policing is part of the network stack. It limits the number of connection requests to the server by using token buckets, which have rate and burst attributes. When those are exceeded the SYN is dropped. The prioritized listen queue allows connections to be organized into pools with different priorities. When a TCP connection is established, the socket is placed into the listen queue according to this priority. URL-based connection control inspects cookies to ID clients to allow content-based connection control.

Because it is difficult to identify specific sets of misbehaving clients, SYN policing, though effective, is in practice difficult to tune correctly. The bucket rates should also be adjusted to the level of resource consumption per request. Simple priority listen queue policies allow lower delay and higher throughput for high-priority connections, but may starve low-priority connections. Combining these two techniques can avoid the starvation problem. In general, kernel-based mechanisms are more efficient than user-level mechanisms. They have the opportunity to toss out the connection before it consumes additional resources.

STORAGE MANAGEMENT FOR WEB PROXIES

Elizabeth Shriver, Eran Gabber, Bell Labs; Lan Huang, SUNY Stony Brook; Christopher A. Stein, Harvard University

Proxies are black boxes with a high-end PC inside, or they're just high-end PCs. In any case, they have particular file system performance characteristics. Files are accessed in their entirety, there is a flat namespace, permission checking is rare, and, since they are caches, proxies exhibit less stringent persistence requirements. The bottom line is that traditional file systems have a lot of unnecessary functionality that is unused and, instead, reduces the performance of proxies. The Hummingbird FS is the response to this observation.

The authors implemented the file system as a library that is linked with the application. The file system and application share the buffer cache. There is no reason to copy data, just pass a pointer. Everything is read/written in clusters, and replacement is LRU. Clusters are files stored together on disk. They are associated via calls that give hints to Hummingbird. A file can be in more than one cluster. Large files are special; they are not cached but are kept on disk. Hummingbird also has parameters that affect the sizes and lifetimes of clusters. The application can specify when and how often to write metadata back to disk.

They implemented it and simulated Squid proxy accesses. File reads were much faster. In most cases throughput improved over five times. Because of clustering, Hummingbird issues fewer disk I/Os than UFS, and recovery is much faster than UFS. It takes 30 seconds for Hummingbird to start servicing requests after a system crash with an 18GB disk. UFS takes 20 minutes in order to `fsck`. The same line of reasoning applies to Web servers as well, though you'll need more functionality, like `ls`, than the toolkit currently provides.

SCHEDULING

Summarized by Joseph Spadavecchia

PRAGMATIC NONBLOCKING SYNCHRONIZATION FOR REALTIME SYSTEMS

Michael Hohmuth and Hermann Härtig, Dresden University of Technology

Hohmuth presented work on pragmatic nonblocking synchronization for realtime systems. Recently there has been a stir about nonblocking data structures. Nonblocking synchronization is useful for realtime systems because it is preemptive everywhere, and there is no priority inversion. It has caught the attention of not only the realtime systems community, but also the operating systems and theoretical groups. In spite of the great interest there are very few known implementations that exploit nonblocking synchronization successfully.

Michael explained that the lack of implementation for nonblocking synchronization is partially hardware related. It is difficult to apply to many modern CPU architectures, because implementation relies on hardware support for atomically updating two independent memory words, such as two-word compare-and-swap (CAS2). For example, the popular x86 CPUs do not support such an instruction.

The authors' work is a pragmatic methodology for creating nonblocking realtime systems that even work on CAS2-less architectures. That is, it does not rely solely on lock-free synchronization. It allows locks, but assures that the system remains wait-free. In addition, the methodology is easy to use because it looks like programming with mutex using monitors.

The authors implemented the Fiasco micro-kernel for the DROPS realtime operating system using their methodology. Fiasco is an implementation of the L4 micro-kernel interface that runs on x86 CPUs. C++ was used to implement the kernel, yet it performs well compared to the original, optimized, non-realtime, assembly language implementation.

The performance evaluation results show that the level of preemptability of the Fiasco micro-kernel is close to that of RTLinux. In fact, the maximal lateness in the Fiasco micro-kernel is an order of magnitude smaller than that for the L4/x86 kernel. This is because L4/x86 disables interrupts throughout the kernel to synchronize access to kernel data structures.

Scalability of Linux Event-Dispatch Mechanisms

Abhishek Chandra, University of Massachusetts, Amherst; David Modberger, HP Labs

Chandra discussed the scalability of Linux event-dispatch mechanisms. Today's Internet servers need to service high incoming-request loads while simultaneously handling a large number of concurrent connections. To handle the workload, servers must employ event-dispatch mechanisms provided by the underlying operating system.

Chandra presented a comparative study of the Linux kernel's event-dispatch mechanisms and their performance measures in terms of dispatch overhead and dispatch throughput. The study showed that POSIX.4 realtime signals (RT signals) are a highly efficient mechanism compared to select() and /dev/poll.

Unfortunately, RT signals have a few drawbacks. They use a signal queue, which can overflow. In such an event, a misbehaving connection can starve other connections from the queue; a different mechanism is needed as a fallback. This may be computationally costly and make applications overly complex. Another drawback to RT signals is that they cannot de-queue multiple signals from the queue simultaneously.

The authors' work includes a solution (signal-per-fd) to RT signals' shortcomings. Signal-per-fd coalesces multiple events and presents them as a single signal to the application. In doing so it also solves the starvation problem by only adding new signals to the signal queue if there is already a signal queued for that fd. Furthermore, it reduces the complexity of the application, removing a need for a fall-back mechanism. Finally, it allows the kernel to return multiple events simultaneously.

An experimental study was done using 252 to 6000 idle connections and 1 byte to 6 KB reply sizes. Results confirm that both RT signals and signal-per-fd have higher throughput, lower CPU usage, and lower response time with many idle connections than do select() and /dev/poll.

Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler

Jason Nieh, Chris Vaill, and Hua Zhong, Columbia University

Vaill presented the Virtual-Time Round-Robin (VTRR) O(1) proportional share scheduler. Proportional share schedulers are useful for dividing scarce resources among users and applications. In proportional share scheduling, a weight is associated with each process. Resources are then divided among processes in amounts proportional to their associated weights.

Early proportional share mechanisms were efficient, but not accurate. One of the oldest proportional share schedulers is Weighted Round Robin (WRR). WRR is an O(1) proportional share scheduler; unfortunately, it is not accurate. This motivated the development of fair queuing algorithms such as Weighted Fair Queuing (WFQ) that provide better accuracy. Unfortunately, in these algorithms the time for selecting a process for execution is a function of the process count.

VTRR is an O(1) proportional share scheduler that is both accurate and efficient. It works by ordering all tasks in the queue by share size. It then allocates one quantum to each task in order, starting with the task with the largest share. Next, VTRR chooses to reset to the first task if the current task has received more than its proportional allocation.

Simulations have shown that VTRR is much more accurate than the WRR proportional share scheduler. On average WRR's error ranges from -398 tu to 479 tu, whereas VTRR's error only ranges from -3.8 tu to 10.6 tu (1 tu is 10 ms). On the other hand, WFQ happens to be more accurate then VTRR; however, VTRR's inaccuracy is on such a small scale that it is below the delay threshold noticeable by most human beings.

Vaill explained that VTRR is simple to implement. It has been implemented in Linux in less than 100 lines of code. For a large numbers of clients, the overhead using VTRR is two orders of magnitude less than the standard Linux scheduler. It is important to note that the Linux scheduler is optimized for interactive tasks, whereas VTRR is not.

The authors performed tests to measure the scheduling behavior of VTRR, WFQ, and the Linux scheduler at a fine time granularity. VTRR and WFQ do a better job of proportional scheduling than the standard Linux scheduler.

In addition, tests with real application workloads (MPEG encoding and running several VMware machines) were performed. In both cases VTRR performs very close to WFQ, trading a very small amount of precision for much lower scheduling overhead. Conversely, the standard Linux scheduler did the worst job in terms of proportional share scheduling.

Invited Talks

Making the Internet Mobile: Lessons from the Wireless Application Protocol

Sandeep Singhal, ReefEdge Inc.

Summarized by Brandon Ching

Singhal spoke on how the Wireless Application Protocol (WAP) can and will meet the needs of the wireless Internet. With the world becoming busier every day and with less time allowed for stationary net access, the growing need for mobile handheld Internet devices is becoming ever more pressing. Along with the demand must come a standard that defines how these devices communicate and work together. That standard is WAP.

WAP is a set of specifications and protocols that explain how things should and must operate while communicating. WAP is similar to TCP/IP in that both are widely accepted standards of how two devices communicate, but with WAP the devices will most likely be cell phones and PDAs instead of workstations and standard Web servers.

Sharing many features with the traditional Internet, WAP allows mobile users to access realtime information easily and gives the added convince of "do it on the go" interaction. This lets you do things such as make flight, hotel, and rental car reservations while at the same time scheduling your 11:00 meeting. Of course WAP also makes stock trading, commerce, voicemail and instant messaging available to you anywhere and anytime.

In case you have been living under a rock for the past five years and insist on asking why bother with all this, Singhal has the answer for you. In terms of corporate and business interests, customer growth and acquisition drive what seems to be an endless push for services and features. And WAP allows a person to keep in touch with family and friends more easily conduct important business and market decision making, and just plain make life a bit more convenient in the process.

All this new technology on the go sounds really nice, so what are the challenges facing this new WAP technology? Since these mobile handheld devices are becoming smaller every day, proper display of information becomes a big problem. The Internet and its services have been designed around traditional PCs, and complex scripting and inefficient HTTP over TCP/IP connections to handheld devices therefore make the acquisition of data and media clumsy. We are also battling over issues such as limited bandwidth and network latency.

The future of WAP looks bright. It has successfully met many challenges, yet it has also fallen short on many Internet expectations. Perhaps the new WAP 2.0 migration to Internet standards will give the performance lift needed for today's unforgiving world.

Evolution of the Internet Core and Edge: IP Wireless Networking

Jim Bound, Nokia Networks; Charles E. Perkins, Nokia Research Center

Summarized by Kartik Gopalan

In this talk, Bound discussed the evolution of IP wireless and mobile computing. He began by observing that the explosion in the number of IP-capable mobile devices, such as cell phones, has placed tremendous pressure on the Internet core infrastructure and edge architecture. The Internet today is characterized by diverse VPNs that have essentially private address spaces and are secure at their edges by use of firewalls, Network Address Translation (NAT), and application level gateway (ALG) mechanisms. In the process, the end-to-end model of the Internet is getting lost. In addition, getting globally routable IPv4 addresses is becoming more and more difficult. For instance, it is virtually impossible for a company to deploy millions of cell phones, each with a globally routable IPv4 address. A solution to this problem is deployment of IPv6, which can restore the end-to-end Internet model and also solve the address space problem. In addition, it enables large-scale deployment of Mobile IP, which is going to revolutionize the Internet.

Bound next discussed the evolution of wireless protocols including GSM, GPRS, UMTS in Europe, CDMA in United States, and finally Mobile IPv6 itself, which promises 2Mbps voice and data over completely IP-based networks. IPv6 is essentially a packet-switching architecture in contrast to today's telephone networks, which are circuit based. One of the challenges is to make IPV6 work in conjunction with SS7, used in today's telephone networks. For instance, IETF's SIGTRAN addresses the transport of packet-based PSTN signaling over IP networks. One of the promising protocols pointed out was the Streaming Control Transport Protocol (SCTP), which enables true streaming that is not possible using present-day TCP.

In order to tackle rapid consumption of IPV4 addresses and routing table explosions, CIDR was proposed as an interim measure. While CIDR reduced the pressure on address space, it still required NAT and ALGs, which imposed tremendous management burden and created a single point of failure in the network. This also imposed performance penalties and prevented deployment of end-to-end technologies such as IPSec.

IPv6, which was standardized in 1998, promises a solution to these problems. It touts 128-bit addresses, has a simple IP header, and is optimized for 64-bit architecture. IPv6 gets over the need for NAT and ALGs and, furthermore, has been designed to be Mobile IP ready. The primary "wireless" advantage of IPV6 is its extended address space. Handoff is complex in IPV6, but it can bypass the triangular routing problem faced in IPv4. Security required during binding updates can be provided by IPSec. Key management will be a major issue in this scenario, for which AAA servers will form the basis.

Bound concluded the talk by touching on the problem of new frequency spectrums that will be required to enable the diverse mobile devices to communicate. He also stated that people from the circuit-switching world will ultimately adapt to this unified communications infrastructure based on IPv6. However, a lot of testing and trials will be required before this actually happens. Jim's prediction was that Asia will be the first to embrace the wireless world since a wired infrastructure is not as enmeshed there as in the United States.

Security Aspects of Napster and Gnutella

Steven M. Bellovin, AT&T Labs – Research

Summarized by Chris Hayner

Bellovin began his talk by describing the many functions common to Napster and Gnutella and, by extension, to every other P2P network. Without central servers controlling the data, the clients are free to decide what to share and what to keep secret. The very protocol supplies the index of peers and connectivity information, allowing direct connection from peer to peer without going through any intermediary.

Napster uses a central server as a base for users to query for files and as a supplier of chat functions. A compiled index keeps track of who has what, at what speed they are connected, etc. By selecting a file of interest, a user gets connection information from the server and then initiates a direct connect to the peer who is sharing the file. Also available is a "hot-list" function, allowing a private list of specific users' connection status.

The Gnutella protocol is different in that there is no central server whatsoever. All users have their own index, which is updated from the indexes of the users they are connected to. This creates a very large network of users connecting to users connecting to users. It is not uncommon for any single user to have up to 10 connections. The Gnutella protocol is an open specification.

The search strength of Gnutella resides in its flooding protocol, wherein a user has the ability to speak to every connected machine. A user searching for a file sends a request to all of his or her neighbors, who in turn forward it to their neighbors. When there is a match, the user directly connects to the user with the file, and download begins. Aside from basic IP address information, there is no authentication of any type.

The talk focused primarily on Gnutella, and at this point, Bellovin discussed at great length the specifics of the Gnutella protocol's five messages: ping, pong, push, query, and query hits. An in-depth discussion of these is beyond the scope of this summary.

Gnutella suffers from the openness of its protocol in several obvious ways. First, there is no authentication of the IP, so the network could conceivably be used in a flooding attack. There would be a lot of attempts to connect to, say, CNN.com, if it were put in a packet that cnn.com:80 was sharing 10,000 files. Also, the Gnutella packet headers contain the MAC address of a computer using Win95/98/NT. This could be used to link requests to requesters and is an obvious privacy violation.

Using a central authority to authenticate makes it very difficult to fake an IP address. The privacy issues are much more apparent here, as the central site could conceivably keep track of every single session for every single user.

The conclusion was that although Gnutella is the wave of the future, there are significant privacy concerns. Authentication of some kind would make the Gnutella network more legitimate as well. Clients need to be well-written to avoid buffer overflows, which are all too prevalent in some kludgy Gnutella clients.

For more information, see *http://www.research.att.com/~smb.*

Security for e-Voting in Public Elections

Avi Rubin, AT&T Labs Research

Summarized by Adam Hupp

With the controversy surrounding our last election there is an increased push to look at needed improvements to our outdated and error-prone voting technologies. Many people are raising the idea of using the Internet for voting, but what kind of risks would that entail? Avi Rubin, who has studied this area extensively, shared his insights and research.

Rubin was invited by the Costa Rican government to investigate the possible usage of electronic voter registration systems in their 1997 election. Voting is mandatory in Costa Rica, and a person must vote in the same district in which he or she first cast a ballot. This creates unique logistical problems the government was hoping to solve with computer systems. Their goal was to register people at any polling site using computers borrowed from schools. Several significant challenges were discovered during the trial. First, the high proportion of computer illiterate persons necessitated the use of light pens instead of mice. Trust was another problem, since the population would not necessarily trust a US-developed system. This was compounded by the fact that US cryptography export laws prevented the use of most encryption systems. In the end, Cost Rica's voting tribunal became worried about challenges to the new system and decided to cancel the trial.

There have been several other groups looking into this issue lately. The NSF hosted an e-voting workshop that brought together technologists, social scientists, election officials, and the US Department of Justice. The workshop concluded that the US is unprepared for remote electronic voting systems but that modernizing poll sites holds promise.

One of the cornerstones of any voting system is voter confidence. There must be confidence that all votes are counted, are counted only once, and remain private. Even as vexed as the most recent US presidential election was, these problems are still more acute in electronic voting systems. Additionally, electronic systems suffer from new problems, such as selective denial of service. What if a subtle denial of service attack was aimed at a carefully picked geographic area? In a close election this could be enough to change the outcome. Trojan horses and viruses pose another significant threat. With the proliferation of this malicious software, how could we trust the integrity of our computers for something as important as a national election?

Cryptographic protocols are a key component of any online voting system. Rubin described a system called "Sensus" developed by Lorrie Craner. Sensus uses blind signatures and a public key infrastructure (PKI) to provide many of the properties of a good voting system. Unfortunately, it is still vulnerable to non-cryptographic attacks, such as a lack of anonymity and denial of service. This illustrates some inherent problems with voting over the Internet.

A longer (2–3 week) voting period to combat the risk of DDoS attacks on voting systems would still not prevent selective denial attacks that subtly reduce service to targeted areas. Additionally, there is always the possibility of large-scale network failures over any time period, which could prevent the election from happening.

Online Privacy: Promise or Peril?

Lorrie Faith Cranor, AT&T Labs-Research

Summarized by Carson Gaspar

Online privacy has now become enough of an issue that it appears in comic strips. Several (rather humorous) examples from Cathy appeared throughout the talk. After the comic start, the talk moved into a brief overview of how private information can be transmitted without the user's explicit consent. "Browser chatter" refers to the extra information sent by Web browsers to Web servers, which includes the IP address, domain, organization, referrer, client platform (OS and browser), the information being requested, and cookies. This information is available to various parties, including the Web server, the server's sysadmin, one or more ISPs, possible third parties such as advertiser networks, and, potentially, to log files that can be subpoenaed. The talk then moved into more specific examples, with a discussion of Web bugs (an invisible image used to gather information) and inappropriate data in referrer headers (such as credit card information). Examples were given from several sites, most of which are now fixed or defunct.

The talk then moved from technology to political and social issues. Various surveys show that people are increasingly concerned about privacy. The European Union has acted on this, issuing a Data Directive that restricts how information can be collected and distributed. The United States has passed the Children's Online Privacy Protection Act and a few other pieces of legislation and industry-specific regulation, but it is far more piecemeal. Data collected by third parties is being subpoenaed increasingly often, both in criminal and in civil cases. The only way to avoid this is to not store the data in the first place.

Some solutions were then discussed. Voluntary privacy policies, privacy seal programs, legislation, corporate chief privacy officers, and client software can all help make things better, but none are a complete solution. The talk then focused on one particular technology: P3P (Platform for Privacy Preferences Project – *http://www.w3.org/P3P*). P3P provides a means for encoding a site's privacy policy into a machine-parseable format, and a means for a client to retrieve that policy and act on it. The server-side tools are available now, and client tools should start appearing by the end of 2001. Microsoft's IE6 beta already includes some minimal P3P support. The open question is will users obtain and use privacy software, even if it's free?

Coming to Grips with Secure DNS

Jim Reid, Nominum Inc.

Summarized by Chris Hayner

Secure DNS, or DNSSec, was developed as a way of validating the data in DNS lookups. The standard, described in RFC 2535, verifies the authentication of DNS responses and prevents spoofing attacks. The protocol uses DSA or RSA cryptography to digitally sign all DNS traffic.

This service, best implemented in BIND 9, does not do anything to stop DoS attacks. There is also the possibility that the DNS server has been compromised, and even though the signatures continue to be correct, the data could be incorrect. It is important to remember that even though secure DNS is implemented, there are still many Internet security holes to consider. The service also does not provide confidentiality of data. This is both because DNS data is public to begin with, and because in some cases an enormous amount of data would have to be encrypted, wasting a lot of time. Thus, only a hash of the DNS resource record is encrypted.

The new keys in a DNS record include: KEY, which represents the public keys of entities named in DNS and is used to distribute keys. SIG is used to authenticate the other resource records in the DNS response. NXT is used to deny the existence of a particular name in a zone. There is also a TTL, or time to live, set for the key and encrypted along with it. This prevents unscrupulous servers from setting unrealistically long TTLs in the plain-text field. Signatures also include a creation time and an invalidation time for keys. Thus, servers with knowledge of absolute time can easily determine if a key is still in effect.

Each zone would ideally be signed by its parent zone, thus creating a chain of trust all the way back to a root server. This leaves us with the obvious problem of where the chain begins. Therefore there is also the option to self-sign a zone, bringing the problem of authentication back onto the field. This is one of the many difficulties in bringing secure DNS into common use.

There is also a protocol called Transaction Signatures, which is a much simpler and much more inexpensive method of securing DNS transactions. It is a very simple protocol, allowing for authentication using shared-secret authentication. This can be used to authenticate responses as coming from the appropriate server. As yet there is no way of distributing the shared secret key.

Active Content: Really Neat Technology or Impending Disaster?

Charlie Kaufman, Iris Associates

Summarized by Chris Hayner

Kaufman opened up his talk with the revelation that the "world's computing and communications infrastructure is a security disaster waiting to happen." To prove his point, Kaufman reminded us that most computers are connected to the Internet in some way, and that these computers have widely known, unpatched security vulnerabilities. He discussed how the animations, CSS, and rich text have anesthetized the masses to this danger.

Active Content is defined as something that is procedural, rather than interpreted data. Email that is simply read is interpreted, while email containing JavaScript and CSS is procedural, requiring a program to run locally to get the information out of the email. Other examples include Java or ActiveX on Web sites and executables and scripts sent and run as attachments.

The current security procedures against such things are very limited. Virus scans only detect known viruses, and firewalls can be avoided by email attachments, etc. Having to track down attackers is tiresome work, and even if the enemy is sighted, he may just be another victim, passing along the bad word. Using a different platform to work in the Internet is a very short-term solution, a solution which also robs users of the user-friendly tools available in Windows development models.

With one mistake, a computer must be assumed compromised and under the control of malicious malcontents. Disconnect from the network and reinstall is the only solution.

The problem has been with us since the beginning. As OSes have become more user friendly, they have naturally become less security conscious. The world was conquered by DOS, a software never meant to be networked to begin with. As the OS has gotten easier to use, the average user has become more naive to the inherent security risks associated with the Internet.

One possible solution is to use sandboxed applications. This would have the program run only what it needs, and prohibit random system calls. This has been implemented in Java. Problems include buggy sandboxing, the legitimate need some programs have for things such as saving state. Also, naïve users may improperly allow programs to override sandbox rule sets.

Having programs signed and authenticated by an authority could allow for security with attachments. This solution is limited by the configuration on either side, and unavailability of the authority for key authentication could cause delays.

Unicode application in browsers has opened a whole window of problems for stopping the execution of malignant code. This character set provides many different ways to say every letter, not all of which will necessarily be interpreted and blocked by the browser. Thus, writing all the possible permutations of a restricted tag could result in its execution.

The ultimate solution is to have OS-level protection from any application overstepping its bounds. Users should have the lowest level of privilege to be productive, and no more. There are applications like sudo for higher privileges.

Myths, Missteps, and Folklore in Protocol Design

Radia Perlman, Sun Microsystems Laboratories

Summarized by Kenneth G. Yocum

Dr. Perlman reminds us that she's going after the sacred cows. The audience giggles, apprehensively. She gives a couple of guidelines: learn from our mistakes, stop making them, and make new ones. She wants to talk about how we got where we are. She starts with "bridges and routers and switches, oh my!" This is because people who think they know the difference between these usually don't, and those that are confused by these terms do.

A brief overview of the ISO OSI reference model is given. Layer 1 is physical, layer 2 is link (neighbor to neighbor), layer 3 is talking across multi-hops, layer 4 is TCP, and layer 5 and above is boring. Everyone laughs. She goes on to highlight the confusion between bridges and routers. She is annoyed at the Infiniband people for leaving out a hop count.

Ethernet muddled everything, and now you need a next hop address in addition to an ultimate destination. So layer 2 source/destination changes with each hop, but layer 3 stays constant. Thus routing algorithm had to be rethought.

She dispels the myth that bridges came before routers. People thought Ethernet replaced layer 3, and they put protocols above it without layer 3. Her boss told her, we need a magic box between two Ethernets. She said, no, we need a router. But they said, no no no. Kludge it in without layer 3. And so the bridge was born. A box that listens to all and forwards everything to the other side. Ethernets can now scale physically, but you need a loop-free topology. Without a hop count, loops in your topology are evil. Evil is defined as exponential proliferation. With routers one packet remains solo. With bridges the packet gets repeated on multiple links, and cacophony ensues. Solution? A clever way to turn off certain links. Radia is clever. She creates a spanning tree algorithm. She reads a poem about it. It is funny. It is good. We laugh.

Radia finishes up with routers. She then moves on to IP multicast. She asks, "How did it get so complicated?" It doesn't have to be hard, she says. ATM had point-to-multipoint virtual circuits. One could add destinations to those virtual circuits. IP people wanted the joint to be initiated by a member, not root. That's OK too. Send a message to the root.

IP multicast API design axiom: it should look like Ethernet – multicast above layer 3 should look like multicast on top of layer 2 (Ethernet). Reality is there's no way to do this efficiently. Address allocation is a nightmare. She lists a variety of techniques: DVMRP, PIM-Dense mode, MOSPF, MSDP, and core-based trees (CBT). She lists the problems. She proposes address of eight bytes. Root of tree is intrinsic part and there are no root candidates. Choose root, ask root for address (root, G). Thus, apparently, addresses are trivial to administer, it's easy for routers to know who the root is, and addresses are plentiful. To quote Hoare: "There are two ways to design software. One way is to make it simple so it's obviously not deficient, the other is to make it so complicated there are no obvious deficiencies." We appreciate the relevance of this quote. She begins to delve into IPv6.

She wonders about the demise of CLNP, which was just like IP but had more addresses. It got killed by IETF when they said you can't replace IP with ISO. "Of course not," she says, "one's a packet format, the other's a standards committee." She dispels more IPv6 myths. It is good. Now she talks about unstable protocols. They are bad.

Example: ARPANET flooding. Because everything was homogeneous, they could find the problem. In about 20 hours. That was then, with 100 routers. Today it would be a huge disaster. Thus unstable protocols are bad, self-stabilizing protocols are good. No one argues. Radia knows much, and her logic is good. We are listening intently. She begins to talk about BGP.

She wonders, "Why isn't there just routing?" We use policy-based routing for inter-domain routing, and cost-based routing for intra-domain routing. But BGP doesn't support all policies. And it supports policies that don't converge, ever. The BGP specification "helps" by saying, "Don't do that."

There are more examples of bad protocol design. SSL version numbers whose field location changes. She argues that simplicity is good. Again, we listen, raptly. She summarizes.

The Internet has to be reliable and self-managing. Protocols have to be simple so that multiple vendor implementations have a hope in hell of working. In the presence of failure, it must at least be self-stabilizing. When you're making a protocol, her advice is, first know the problem you're trying to solve. She tells a story to illustrate this point. We love her stories. One day her child was crying in the hallway, holding his hand. She ran over, and said, "Everything will be OK!" kissing his hand to make it better. She asked, "What happened?" He said, "I got pee on my hand." Everyone laughs. People whistle and cheer. It is stupendous.

There are questions. The best one is, "Do you have any more stories?"

Strangely Enough, It All Turns Out Well (Adventures in Venture-Backed Startups and Microsoft Acquisitions)

Stephen R. Walli, Microsoft Corp.

Summarized by William R. Dieter

Walli discussed lessons he learned during the birth, development, and eventual acquisition of Softway Systems, a company he co-founded in 1995. He said the most important factor to the success of a startup is passion for the product. The founders must believe in the product. The book *Silicon Valley Way*, by Elton Sherwin, mirrors Walli's experience at Softway.

Softway started out with the idea of making POSIX compatibility work on NT. With just one person on the payroll and the other founders working other jobs to pay the bills, Softway met its first deadline in March of 1996. Despite front page press coverage at Uniforum that year, bootstrap funds were running out. In its first round of venture capital, Softway took $2.2 million, even though it was offered $5 million, because the founders wanted to retain control of the company. Walli believes not taking more money was a mistake. He said a company has big problems if the founders have to use their stock percentage to win votes on the board of directors. Later, the executive team found themselves spending more time raising money than running the company because of this mistake in both the first and second round of funding.

After the first round of funding, the company continued to grow, and it gained acceptance from some early adopters. By that time, the company of about 27 people was ready to make the leap to mainstream acceptance, as described in *Crossing the Chasm*, by Geoffrey Moore. One issue that Softway's management team did not fully understand was that to cross over the company had to do everything possible to get one big mainstream customer even if it meant neglecting other customers. Many large companies will not commit their business to a new product from a small company until they see other mainstream companies doing it. It is difficult to ignore smaller customers who are willing to pay for the product and who have been loyal in the early adopter phase, but who will not help win mainstream acceptance. In addition, it is crucial that employees know the company's goals so they can explain them to customers. Though Softway eventually got a deal with Dell to ship their product on every machine sold to the U.S. government, they were not able to get enough big mainstream customers to stay afloat.

By November of 1998 Softway had grown to around 40 people. Though Softway was bringing in around $2 million per year, it was still not profitable and money was drying up. When the cash started running out Softway had to lay people off. Layoffs are difficult for a startup because the management team often knows and has worked closely with those who are losing their jobs. Softway hired a banker to try to find a buyer for Softway. *Five Frogs on a Log*, by Mark Feldman and Michael Spratt, discusses what works and what can go wrong in mergers and acquisitions.

After prolonged negotiations that came tantalizingly close with several different companies, Microsoft agreed to buy Softway. All of the employees except for the executive team had to go through a hostile interview for a position at Microsoft. The Microsoft interview procedure is designed to hire only the best employees who will fit into the Microsoft culture. Microsoft's goal is to hire people who will be good for Microsoft first and for the particular position second. As part of Microsoft, Walli and former Softway employees had to adjust to the Microsoft culture detailed in *The 12 Simple Secrets of Microsoft Management*, by David Thielen.

If he had it to do again, Walli said he would take more money sooner because a little stock that's worth a lot is better than a lot of stock worth nothing. He would also be more particular when hiring and focus on "crossing the chasm." It is important to keep everyone focused on the company's mission. Walli said that he would "do it again in a heartbeat" if he found another product for which he had the same passion.

Several questioners wondered how Microsoft deals with employees who have made enough money not to worry about getting fired or raises. Walli replied that the Microsoft culture breeds relentless motion. Lower-level employees are driven by compensation that is closely tied to performance reviews. Those who are fully vested and no longer want to work generally quit and make way for those who are lower down on the ladder. David Thielen's book describes the process in more detail.

The Future of Virtual Machines: A VMware Perspective

Ed Bugnion, VMware Inc.

Summarized by Kartik Gopalan

Bugnion presented the state-of-the-art and future trends in Virtual Machine technology. He began by giving a historical perspective on virtual machines. The IBM mainframes in the 1960s and 1970s, such as IBM VM/370, were expensive and hence were designed with virtualization in mind in order to support efficient use of system resources. In the 1980s, as the desktop PC revolution began, hardware became cheaper and diverse, and the concept of virtualization was forgotten for a while. In the 1990s, Mendel Rosenblum, Ed Bugnion, and others began the Disco project, which aimed at running multiple commodity operating systems on multiprocessor MIPS machines. The project was named Disco since, at the time, virtualization was thought to be just another bad idea, like the disco music from the '70s. However, with budding interest in this technology, VMware took its present shape.

The principal challenges faced by the virtual machine concept were virtualization of the most prevalent IA-32 architecture, the diversity of present-day hardware, and acceptance of the idea by users. The result is the VMware workstation, which has the look and feel of a regular user-level application, and the VMware server, which has a Web-based management interface and remote console facility. Essentially, VMware provides an additional level of indirection between the operating system and the hardware, thus enabling the coexistence of "multiple worlds." A world consists of the OS, applications, and associated libraries.

The basic requirement of VMware is that the CPU needs to be virtualizable. Accordingly, CPU architectures can be classified as "strictly virtualizable" (such as Intel Alpha and Power PC) and "not strictly virtualizable" (such as IA-32 and IA-64). It is the latter class that is the most challenging but also the most useful in present-day scenarios.

The hosted VMware architecture allows a guest OS to execute on a machine where a host OS is already installed and running – for example, allowing Linux to run within a Windows NT environment. Advantages of this architecture are that the guest OS behaves as if it is just another application running on the host OS, the implementation is portable, and it works in the presence of other applications. However, it is limited by the scheduling decisions and resource management policies of the host OS, and it incurs heavy performance overheads due to world switches and during I/O accesses. One of the challenges in this architecture is virtualizing hardware, i.e., supporting any number of virtual devices.

The VMware ESX server architecture eliminates the need for a host OS. It is a micro-kernel-based architecture with a thin VM kernel sitting above the hardware and multiplexing hardware accesses by multiple guest OSes. The principal advantage of this approach is high performance I/O. It also opens up opportunities of customized resource management policies for each guest OS.

Some of the usage scenarios include testing and deployment of new software, server consolidation, allowing applications from multiple worlds to coexist on the same hardware platform, and security. One of the predicted future trends is that virtualization will have an impact on processor architecture and hardware designs. There will be more pressure to build designs that are easily virtualizable with minimum overheads, especially due to trends toward bigger servers and server consolidation. Virtualization, it is also predicted, will impact system software. Many problems, such as performance isolation, are better solved below the operating system. Further, operating systems will be optimized to run with VM, and there's a possibility that device drivers will be written for idealized devices rather than diverse real hardware. This would also allow new innovations in operating systems to take shape quickly and not be bogged down by hardware diversity. And the trend toward building compute clusters based on virtual machines and virtual storage would gain momentum.

Closing Session

The Art and Science of Sociable Machines

Dr. Cynthia Breazeal, MIT Media Lab

Summarized by Jon Stoffel

Dr. Breazeal's closing talk was a fascinating look at how humans and robots can interact, and the ideas behind this interaction.

She started off with a quick survey of robots in film, and how, since the 1950s, they have evolved from single use, barely humanoid robots, into more complex, interactive robots, evolving from HAL to C3PO to Data. She then showed a video of the Sony stand-alone robot SDR doing aerobics, dancing, and kung fu moves that showed how the autonomous state-of-the-art had advanced recently.

The core of the talk was about Kismet, a robotic infant designed by the Sociable Machines Project at MIT. Breazeal gave a quick history of autonomous robots which mirrored the evolution of science fiction robots. The Mars surveyor worked in a slow-changing environment, was isolated, had limited contact with us or other robots, and had pre-specified tasks with strictly limited autonomy. RoboCup, a robot soccer league under development, involves a rapidly changing environment, robots that are autonomous but have to work in teams, and very specified tasks. Humanoid interactive robots will need to work in a very complex environment, perform open-ended tasks, be very autonomous, and interact in a complex manner.

The Sociable Machines Project decided to use an "infant caregiver" metaphor for their investigations. This was a change in the standard assumptions for the training environment of robots. The idea was to build the set of constraints from interacting with people, not pre-programming.

Some of the issues involved with a human-centric robot include deciding what matters, deciding when to try to solve a task, evaluating the results of an action, correcting improper actions, recognizing success, and structured learning.

Kismet is the result of their work. It combines elements of the appearance and personality of a human. The robot itself is just a head and neck on a box, but it mimics human child qualities of cuteness by portraying innocence, youth, and curiosity. Kismet is highly expressive, with lips, eyebrows, and big, mobile eyes.

During the talk, we saw several videos of Kismet interacting with women of all ages, from kids to adults. These videos can be found on their Web site (see below).

The interactions demonstrated various areas of perceptual and expressive systems that had been developed in Kismet. These included visual recognition algorithms which were implemented to include such features as "looking" preferences. Sometime Kismet would concentrate on the person's face, at other times it would search for and concentrate on the object being waved at it.

Kismet was also programmed to recognize "vocal affective intent" when people spoke. It was very funny and interesting to see how people used the visual feedback of the robot's shape to drop into baby talk when interacting with Kismet. Kismet was able to recognize and respond to various types of vocal noises including: soothing, attentive, and prohibitive.

A third area was Kismet's emotional systems, the expressive feedback that Kismet gave the user. To keep up the performance, the software consisted of small self-contained modules that were chained together. Kismet generates expressions in a virtual 3-D space, which it then uses to drive its response. The space includes axes of arousal/sleep, calm/excitement, and stress/depression.

Fourth, Kismet's emotive voice gave the user audible feedback. This was driven by the DECtalk speech system. The audience laughed at the disgusted and sad samples that were played.

The highlights of the talk were several videos of Kismet which pulled together all of these subsystems into a whole. They included, for example, Kismet's visual interactions and preferences, described above, and Kismet's being scolded (in German even!) until it would lower it's eyes and look downcast. These were very amazing for their life-like feel; you started to forget on some levels that Kismet really was just a robot.

Breazeal then concluded her talk with a summary of where we are now and where future work needs to be done. All in all, this was a fascinating talk. For more information, visit *http://www.ai.mit.edu/projects/kismet.*

USENIX Quiz Show

Summarized by Josh Simon

As usual, Rob Kolstad closed the conference with another rousing Quiz Show. With all-new categories and topics this year (most of which were written on Saturday), the audience and contestants once again enjoyed themselves.

The contestants and scores were:

Group 1 Christopher Davis (3500), Steve McIntyre (2900), Perry Metzger (900)

Group 2 Andy Tannenbaum (2100), Mark Langston (2000), Matt Crosby (700)

Group 3 Ethan Miller (3500), Jim Larson (2900), Michael Buselli (1400)

In the finals:

Ethan Miller (5700) Christopher Davis (1700) Andy Tannenbaum (1300)

And in the Tournament of Champions:

Aaron Mandel (2100) Ethan Miller (2100) Trey Harris (1900)

In the tie-breaker Aaron scored 500 and Ethan scored 1000 to be the grand winner.

The USENIX Quiz Show has been produced by Rob Kolstad, Dan Klein, Dave Parter, and Josh Simon. Testers were Rik Farrow and Greg Rose. Special thanks to MSI for audio-video assistance. Prizes were provided by USENIX, Radware, O'Reilly, Prentice Hall, Addison-Wesley, ActiveState, Tandberg, SEI/CERT, and Integrated Computer Solutions. This has been a Klein/Kolstad Production. Copyright (c) 2001.

Photo Galleries

Several photo galleries of events at USENIX 2001 can be found at *http://www.usenix.org/publications/library/proceedings/usenix01/photos.html.*

;login: welcomes submissions of photographs of USENIX events. Send us the URL for your particular gallery.

needles in the craystack: when machines get sick

Part 7: Diagnosis – A Projection of LISA to Come?

by Mark Burgess

Mark is an associate professor at Oslo College and is the program chair for LISA 2001.

Mark.Burgess@iu.hio.no

And now remains
That we find out the cause of this effect
Or rather say, the cause of this defect,
For this effect defective comes by cause.
(Hamlet, 2.2.100–4)

Earlier in the series, I talked about how computer systems can be understood in a framework which befits any complex, dynamical system, by viewing changes as signals (i.e., processes) which compete for dominance in complex environments of many players. I talked about how order has a price and how disorder or uncertainty inevitably grows, unless it can be held in check by an idealistic, ordering "potential." I discussed how human attitudes complicate matters by fixing expectations, demanding policy, over-simplifying evidence and thus losing important information, by complacency, and even by irrational psycho-social instinct.

One might get the impression from all of this that the situation for understanding and stabilizing computer systems is rather hopeless, that system administration is really a "soft" subject with no hope of rational analysis. My reason for embarking upon this series is that I believe that this is too pessimistic a view. Looking around at the world we live in, there is astonishing order, in spite of the odds. It is my suspicion that the main limitation in our understanding, is not the world of computers, but rather our vision of them.

How then can we go beyond bemoaning our troubles and come to firm conclusions about improving that understanding? Aiming to do science, rather than guesswork, we need to formalize our methods and investigations and erect a framework for study which is both criticizable and refinable – in which it is possible to *know*, within quantifiable tolerances. Fortunately, the ideas in the previous chapters of this series hold the answers.

The most fundamental and profound of all principles in science is the principle of causality:

For every effect, there is a cause which precedes it.

(See my book *Principles of Network and System Administration*, published by J. Wiley & Sons, for a further discussion of this.) Causality, framed as information theory, was the thrust of Part 5. It might seem trivial, even obvious, but this foundation of all change is quickly forgotten, even by scientists and engineers, when the going gets tough. As I noted in part 5, causal influence is a mapping from events which occurred in the past to events which are occurring now. It is an N:M mapping, i.e., a many-to-many mapping. Each observable phenomenon stems, in general, from several causes, and, conversely, each causal factor leads to many consequences. This is what makes matters

Every security issue essentially boils down to a problem of whom or what we are willing to trust

hard to unravel. When we make observations of the present, it is impossible to say with certainty what the cause was. The best we can do is to see whether statistical evidence supports a model or hypothesis. So the central problem becomes: how do we formulate such a model?

We can study systems empirically and obtain clues, but empirical studies have many shortcomings. What is needed is a simplification. The aim of science, after all, is to provide *suitably idealized descriptions* of phenomena, so that they may be analyzed and verified to within the limits of their assumptions. Science is not about complete descriptions, with every detail pinned down. The latter would be impossible, since the level of detail in the perturbing environment is essentially infinite.

What about the human aspect? Sometimes colleagues suggest to me that one cannot apply science to problems like human management. I find this astonishing. Management is nothing more than the problem of scheduling of resources in space and time, given a somewhat fickle environment. Although science will not have the exact answers, because it is *always* about simplification, the idea that one would rather revert to witchcraft than surrender a problem for analysis is rather frightening. Either such colleagues have no faith in science (in which case they are just bureaucrats going through some learned motions, and will never find anything new), or they are blinkered into believing that knowledge is devoid of principles which can be applied beyond an immediate context.

Clearly, computer science has a lot to say about how data structures and scheduling algorithms can be applied. If they can be applied to computer programs, they can be applied to humans. The results will not be exactly the same, nor exactly predictable, but it is possible to make the study and learn something.

Security is an excellent demonstration of causal trees. Every security issue essentially boils down to a problem of whom or what we are willing to trust. Every security problem can be drawn as a causal tree. At the top is the thing we want to secure, it splits into everything that thing depends on, then in turn the dependencies of each of those elements, and when we decide to stop this (at some arbitrary level of recursion) we end up with a number of possible sources of security breaches. Those are the things we are placing our trust in. If we don't like some of them, they can be replaced by other things, by putting some technology in the way, making another link, and moving the trust. But, however we look at it, we cannot escape this causal dependency. Security hangs on the threads of trust.

Cause-Signal-Effect and Projective Digitization

To sum up the series so far, science can be understood as a causal analysis. Such an analysis needs a motivation, or a direction which can be used to trace the tangled skein from cause to each effect. This is the role of a model. Without it, one is immediately confounded by multiplicity: many causes have many consequences. We have to be able to separate the interesting signals from the background noise.

If you have been following the series, you will now realize that we know something about this problem. It is just information theory: the theory of signals. All causal phenomena can be discussed in terms of the theory of communication, because the arrow of causal development can always be mapped onto the basic idea of a signal from past to present, or cause to effect. Some signals are strong and obvious, while others are down there amidst the noise.

Causal analysis of a system's behavior is also the skill of diagnostics: it is a systematic and logical imitation of the evolutionary probing which complex environments exert on systems. While the involuntary complexity of environment alone will get you sick, a doctor has to simulate complexity systematically by prodding and asking: tell me when it hurts. Tracing backwards from effect to cause is only possible if the mapping is one-to-one, and the information about changes is preserved. One-to-one mappings only occur in strictly isolated systems, with stringent, reversible protocols. Such things are rare, as it happens, and usually only possible for infinitesimal changes, because larger changes inevitably convolute with the environment.

The human computer interaction is fraught with much error

Perhaps you are still of the belief that the relationship between cause and effect is a simple one, that we can just decide how systems should be, introduce "management," and bingo! If so, it is already clear that you are not a perfect manager, but I ask: are you a perfect typist? Consider your interaction with the keyboard as an input device. The human computer interaction is fraught with much error. The interface itself is digital. When we hit the space bar, we do so with information about exactly where we hit it, how hard, how fast, and so on. That might be affected by muscle spasms, distractions, or (in the case of my laptop) random electromagnetic spikes. The computer digitizes this into the coarse classification space or no space. It is a many-to-one map. Information is lost and cannot be recovered.

Now suppose we try to hit the "M" key: now there is a finite chance that we might actually hit the space bar, or the "N" key. Again, the reason for this is lost to the computer, but the result is not: it is neatly classified and recorded, giving a precise yet wrong outcome. The effect is said to be projected into the space of outcomes, which is digital. Determining the cause of a bad key hit is so difficult that most would call it a waste of time to try, but it happens quite regularly, because between the brain and the CPU, there is a bunch of environmental contamination: what Shannon would have called a noisy channel.

An almost identical case of projective causality is found in the hierarchical form of evolution. Phylogenetic trees are branchings of species, which record the causal influence of an environment, projected onto digital genes. Although the tree provides a simple relationship between previous and current, it is a projective description, like a string of typed characters, full of errors. It has forgotten all of the environmental information which led to the changes. It cannot be "rolled back."

Digitization leads to a *projective* representation, like the shadow of an object on a wall, the impression left on the keyboard, or on system resources. Part of the information is dropped, and only an impression of the truth is left as a clue to what really happened.

Causal Trees with Imperfect Information

Computer scientists have acyclic, directed graphs growing in their gardens. That is the graph-theoretical name for a tree. Tree structures abound in science of all kinds, because they are direct representations of causality. In an ideal *microscopic* description of a system, we would know every detail of every change and be able to trace each one from cause to effect in a huge complicated tree. In order to draw such a tree, we would have to have *perfect information* about the changes in the system.

Because data are projected onto a finite, digital map of resources, some of the causal branches which should be there are missing, lumped together. This means that there is hidden information in the projective paths. If temperature of the machine room could

Human issues like customer satisfaction can play a role in system administration. Ideally, it would be possible to eliminate such subjectivities, but users have an irrational insistence on their own subjective wishes

affect the results of transactions, then that information would also have to be measured and recorded, to get the full picture; if stray cosmic rays could affect input, the results of transactions (as they do on my laptop's electrostatic mouse), then they would also have to be catalogued. Since these things are not recorded in the workings of the machine, a probabilistic element enters into the projective result. Perfect information is stifled by projection. Some administrators try to achieve it with auditing, but even the molasses of information in full system accounting are never complete, because nothing on the system can record what motivates users to do what they do.

When addressing complexity, one does not normally pretend that exact results are possible; rather, one tries to model probable outcomes of the system. Such an analysis must have "hidden variables," which represent what is not known about the system. The best one can then do is to look for likely or possible outcomes using a causal tree analysis.

One kind of analysis is "risk analysis." Risk analyses are common in a variety of disciplines and go by many names (see Rob Apthorpe's paper at LISA 2001 for an application of the method to system administration). Such analyses usually attempt to quantify the different causal pathways in terms of some idealized reward called "payoff." Risk can be minimized, profits can be maximized, "uptime" can be maximized, and so forth. How the payoff is defined depends on what one is interested in achieving. It is essentially a matter of policy.

Framed as a principle for minimizing risk or maximizing payoff, the optimization problem is one of extremizing a parameterized function. This is something which is well known in the sciences: the principle of minimum risk, the principle of least action, Fermat's principle, the minimax principle. All of these are variational methods looking to optimize some criterion. It is essentially a search-algorithm for probing the parameter space of possibilities for a desirable property.

The properties one might hope to maximize or minimize represent desirable or undesirable pathways from cause to effect. Risk, productivity, user satisfaction, return on investment, etc., are all abstract qualities which are baked into the causal pathways with probabilities that arise from the hidden variables. In contrast to many other areas of analysis, such as pure economics, artificial intelligence decision-making, human issues like customer satisfaction can play a role in system administration. Ideally, it would be possible to eliminate such subjectivities, but users have an irrational insistence on their own subjective wishes. Nothing new there: we are basically concerned with ourselves, not the abstract vagaries of "the system."

States and Models of Change

Our aim is to model how computer systems change by traversing the pathways of a projective causal tree, i.e., we are looking for their dynamical properties, projected onto the set of variables and resources which pertain to the interaction with users. Changes can occur in a machine at several levels; the smallest, most primitive changes (executed instructions, read/write operations, etc.) are often called *microscopic* and happen all the time, over very short intervals. Long-term changes (amount of free memory, level of activity) are called *macroscopic*, because they represent the cumulative effect of many microscopic transactions. Their changes are average changes, and these happen more gradually since there is some reinforcement and some cancellation of the microscopic changes over time.

What variables characterize the system? Are they functions of time, continuous (smooth) averages or discrete (digital) measurements? Software metrics, such as numbers of processes, numbers of conversations, rate of packets per second, amount of free memory all characterize the resource usage of the system, and many more. These reflect changes taking place, but clearly they do not record why, so there must be hidden variables.

One can choose to examine these over intervals of time (micro or milliseconds) during which they change only slightly, or over longer periods (minutes to weeks) which more closely reflect the activity of external influences such as user behavior. In order to build a model, and find answers, we need to compare values at different times. Sometimes it makes more sense to compare changes to the system with a corresponding value measured a few moments before, and other times it will make more sense to compare to a value from a similar time one or more days or weeks ago. As we shall see below, the working week plays an important role in modeling.

A useful, if somewhat overused notion is that of *state*. A *microstate* is a set of values which characterizes the system at some moment. For instance, the simplest dynamical systems, studied in physics are particles which fly around. Particles are characterized by variables such as their mass, their charge, their position and their velocity. This set forms a state of a microscopic element of the system, or microstate. Once we put together more complex, composite systems, we can talk about emergent properties also as describing *macrostate*: for instance, temperature, pressure, roughness, viscosity, etc.

Computer systems are a bit like this; they have primitive things going on, such as atomic operations: read, write, add, locate. At a higher level, we also combine these actions into programs, processes and other structures, which have emergent properties like "busy," "idle," "thrashing," and so on. At the microscopic level, the state of a system can be thought of as the values of a long line of bits and bit operations. At a higher level, one can talk about numbers of processes, user sessions, protocol states, which are coded into the bits at a higher level.

A characteristic of complexity in a system is that there is no unique way of describing it

A characteristic of complexity in a system is that there is no unique way of describing it. Any convenient modeling projection will do, but there is a trade-off. The more detailed one gets, the more information one sees; but information is noise, and meaning is difficult to find. Alternatively, one can step back and perform the half-closed-eye test: there is less information, but the structure is seen more clearly.

What pays, in general, is an approach in terms of the most convenient measurable parameters over the time-scale which is germane to the problem. For system administrators, the variables and time-scales generally occupy the level of the operating system's interaction with users (processes, files, over minutes or weeks). What we are looking for, then, is a description which captures changes of state variables at some arbitrary level.

Markov Chain

The essence of describing such changes in state, is the Markov chain, or Nth-order Markov model, and its more realistic extension, the "hidden Markov model."

Put succinctly, a Markov model is a model in which the state of the system after the next time step depends only on the state of the system now. It is literally a sequence of links in a chain. For example, a traffic light has this property: when the light is red, you

When a system has fairly regular behavior and is affected by hidden variables, it is not completely predictable

know the next state will be green (in the US; red and amber in parts of Europe). When it is green, you know the next state will be amber. When it is amber, you know the next state will be red, and so on. One does not have to remember the entire history of what happened to the traffic light in order to understand what it is going to do next. Markov models are the simplest kinds of model, but surprisingly they describe many situations fairly well. One finds simple Markov models in computer science, but, in this form, they are usually trivial.

Traffic lights and other Markov processes are sometimes said to be in a steady-state, because their behavior is predictable for all time. It doesn't vary. Either it is constant, or it goes 'round and 'round in a *limit cycle*. Alas, not many problems are really quite so simple. Nth-order Markov models are models where the next transition to a new state is governed by the last N states of the system. Such models are sometimes useful for parsing simple grammars but are not very useful for understanding anything as complex as a computer system. A Markov model can be represented simply as a *transition function*, which is a list of now-states and next-states.

Real-world problems are too difficult to solve with this kind of approach. Why? Because the level at which Markov models could be applied is usually so low that the amount of detail would be overwhelming, and therefore simply noise. Instead, one purposely hides some of the data, using the half-closed-eye method, and by making the fundamental separation into system plus environment.

Billiards is a game which is often used to illustrate problems in dynamics. It cannot easily be represented as a Markov model, because the positions of the other balls in relationship to the environment of the table influence the outcome of the next move. In other words, the billiards "system" has a memory of what went on before, and the shape of the table play a role in determining what can or will transpire next. Moreover, there is an external entity in the game: the player. The player brings additional information to bear, which is not on display on the table. Chess is another example, which is digital, like a computer. The state of a game of chess is the position of all of the pieces on the board at a given time. The next move is determined not only by the positions of all of the pieces on the board, but also by the choice of the player. The next move has not one but several possibilities, and the extra information which decides which possible it has chosen is hidden from view. The transition diagram for chess is not one-to-one; there are many possible moves at each stage. The game eventually converges to a checkmate when the game runs into a part of its state-space which is a dead end (checkmate) or a limit cycle (stalemate).

When a system has fairly regular behavior and is affected by hidden variables, it is not completely predictable. A useful approach to understanding it is to look at its average or expected behavior. The average behavior is defined by the mean value of the state of the system over an ensemble of equivalent situations. Each equivalent observation of the system brings new values but, over time, these yield approximately the same result, up to smaller corrections. One says that the system exhibits microscopic fluctuations about its macroscopic average value. The separation

signal = average + fluctuation

is deeply connected to the fundamental split:

world = system + environment

This is not so much a fundamental property of nature, as it is a management viewpoint. This is the way the human cognitive apparatus analyzes: what we expect versus what we see.

Models which describe state transitions with imperfect information are called *hidden Markov models*. There are two ways to handle these models. One is to actually model the external information; the other is to create a *stochastic model*, i.e., one which only predicts the probability that a transition between states will be made. These models will form the substance of models of computers as dynamical systems (see papers by Apthorpe and Haugerud at LISA 2001), since computers have external players called *users*. Hidden Markov models are characterized by probabilistic transition functions, with hidden variables H, e.g.,

$$(s_1|s_2) = P_{12}(H)$$

denoting a transition from a microstate s_1 to a microstate s_2, with probability $P_{12}(H)$, which depends on the hidden variables. The approach has been used to build quite convincing simulations and mathematical models of the behavior of computers in projective representations (numbers of processes, numbers of users, etc). Given such a model, with predictive power, one knows enough about the system in order to characterize its long-term behavior in terms of what is predictable and what is unpredictable. This leads to great simplification and time saving when looking for anomalous behavior.

Boundary Conditions

Every manageable dynamical system makes contact with its environment at some time or place, either at the outset of its evolution or during the act of measurement, at an edge, or at some boundary or interface. The environment leaves its projected imprint on the system: incomplete information about its state. The effect of the environment is usually strong, because the environment is bigger and more pervasive than most systems.

Computers touch base with users via the keyboard and via the network. These channels link computers to a reservoir of thoughts and activity which have a direct impact on what computers do. It would be bizarre indeed if it were not possible to see these effects reflected in the state of the system. Indeed, the working week is easily identified in the patterns of resource usage. It shows a fundamental periodicity in computer behavior, which has its origin in the approximately cyclic behavior of "the average user."

One way to take account of the approximate periodicity is to formulate computer activity as a process on a circular topology (see Figure 1). By winding the time parameter around a cylinder of one-week circumference, and then squashing the resulting spiral into a circle, one ends up with many recorded values for the time-series variables at each point, a bit like old recording weather barometers. By averaging the many values at each time of the week, one then sees average behavior in relation to the working week. This is a more useful description than an average

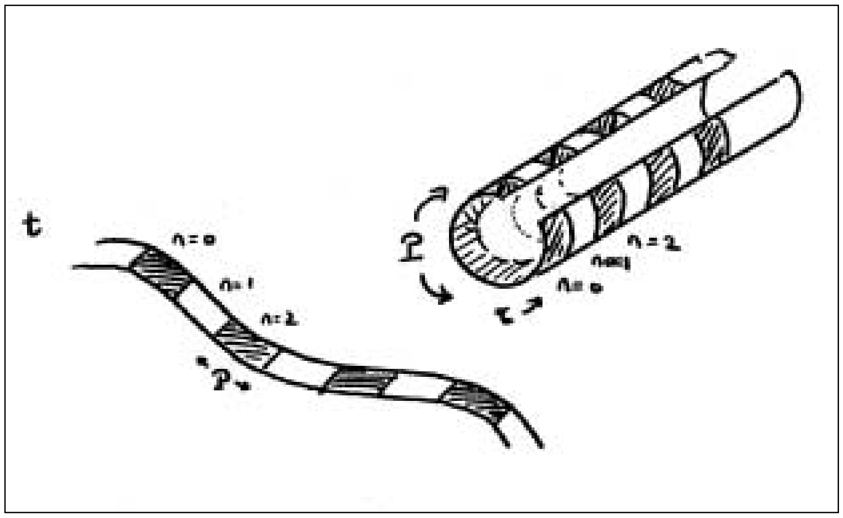

Figure 1

A type I model is a description of the state of a computer, over time, subject to the external behavior of users

over all times, since it contains real information about the changes going on in the environment.

By using stochastic methods or Monte Carlo simulation techniques, on a pseudo-periodic background, it is possible to put together a simple model which reproduces the main features of computer behavior (see the work at *http://www.iu.hio.no/SystemAdmin/scisa.html*). This allows one to say with quantifiable (calculable) certainty when a computer is behaving in one way or another. Any anomalies which are then observed must be due to effects which were not considered by the model and can be singled out as worthy of attention.

We can say two things, at the macroscopic level, about users' effect on computers:

- It is pseudo-periodic (driven by the working week).
- It is stochastic.

At the simplest level of approximation, one could say that the behavior of users was a sinusoidal, diurnal rhythm. This is not a very good approximation, but it is better than assuming that user behavior is constant, as many intrusion detection systems do. With further refinement, one can reproduce the graphs of user behavior displayed (see Haugerud and Straumsnes' paper at LISA 2001, based on our earlier studies), in order to discover how the actual pattern arises, whether it is coincidence or predictable. These patterns are the most basic "laws of nature" in system administration.

I refer to this kind of model as a type I model of a computing system. A type I model is a description of the state of a computer, over time, subject to the external behavior of users. Such a model might have many interesting features: steady-state behavior, dead ends (crashes, deadlocks etc.), and even chaos. I have spent some years working on the separation of system and environment in approximately steady systems (in physics as well as computers) and have a detailed stochastic model in the limit of large numbers of data, which identifies the important scaling properties of the system. To the trained eye, the model is very simple, but it fits the data surprisingly well.

The importance of such a model is in understanding how the structure of *cause* relates to the shape of *effect* in real, observed behavior. While we have barely scratched the surface in our work at Oslo, the results promise to explain many features of observed behavior and can be fed back into actual methodologies and tools such as cfengine. Only when armed with such knowledge does it makes sense to speak of anomaly detection.

Policy Constraints: Type II Models

Type I models are likely to be important as a general guide to understanding how cause and effect are related in computers, but the success of that approach is dependant on how well one can represent the behavior of users, who represent the largest perturbation. It is not just about projecting the world onto a model, it is also about how many nuances the model should cover. Type I models treat users as a relatively formless gas of influence in which no overriding, strong signals dominate. This is a beginning, but it will not be sufficient to deal with real systems, in which a single user can make his or her influence felt by all the rest.

So how shall we know the shape of users' behavior? What happens when users do not obey simple rules, i.e., when they are not a formless gas, but an obelisk: a needle in the Craystack? Is it still possible to gauge their effect on the system? The answer is yes, though the difficulty of doing so steps up an order of magnitude. The reason is that crowds of users behave in simpler average patterns than individuals, just as a view from a distance looks simpler than a view in close-up. Crowds have a natural inertia in number: the averages are augmented only by small fluctuations. However, in smaller group sizes, fluctuations can dominate over the average part, leaving a view of disorder.

The success of science is largely based on the idea that the laws of nature are constant, and that one therefore stands a fighting chance of unraveling them. If the rules are changing too fast, one cannot find meaning in the variation, and the good goes from bad to ugly. One thing one can do then is to look at the long-term variations only, by averaging, and find laws for those. As I said at my LISA 2000 talk on "Theoretical System Administration," there are no "Newton's laws" of system administration. There is no single set of rules which governs right from wrong, likely from unlikely. Why not? Because each site has its own environment and its own policy for dealing with it. Strong individuals will shine against this background.

The success of science is largely based on the idea that the laws of nature are constant, and that one therefore stands a fighting chance of unraveling them

Policy can be used to evaluate user behavior numerically, by defining a scale of value. The value is "payoff" once again, only now one must also say, payoff from whose perspective? The scale is not necessarily unique. It is only required to be consistent in all comparisons. The idea of scales of values determining social behavior is a fascinating problem which has plagued the social sciences for many years. What is new and interesting about computers is that we actually have a chance of quantifying the behavior stringently, because the machine can see everything that is going on, within an automatically limited arena. Also, formalized value systems can be evaluated impartially.

Our quest, then, is to evaluate the likely mixture of behaviors in a mass of users according to some criterion. The scale of measurement will be related to system policy in the sense that users will tend to aggregate around behaviors which are provoked by what they are allowed or supposed to do. Some users are law-abiding or altruistic; others are contrary and selfish. Mixed up in here, is the somewhat fluid notion of "security"; it is rather hard to pin down, but it is clearly related to the extent to which the system and its users work within the boundaries of policy, and the idea that an unfortunate mixture of user behavior might drive the system into an undesirable state.

A model which evaluates a profile of user behavior in relation to policy is what I call a type II model of a computer system. Such a model is not completely independent of type I models. Rather, the two feed off one another.

Policy and State: Paths through a Lattice

At the most primitive level, the resources of a computer can be thought of as a string of bits, subject to external change: disks and memory are represented by the bits, and the external change comes from I/O with users and the network, mediated by the CPU. The structure that we build on top of this bit string, including the file system, the operating system, the structure of data, and so on, is multidimensional, and discrete, i.e., it forms a lattice. As we look at changes in the system, we can classify those changes on this lattice. The contention is that, when one decides policy, the effect is to select a preferred region of this lattice. In other words, policy is a projective action, which effectively selects one or more acceptable regions of the state space.

As far as a computer is concerned, the effect of a system policy is to do the following:

- System: specify machine configuration in terms of allowed behavior, access controls etc.
- Environment: encourage users to obey limitations and work patterns.

The initial configuration of the system, places it within a region of the lattice which is chosen by policy. This is controllable and verifiable. Asking users to obey rules is politely asking them not to try to drive the system away from this policy region. This is not controllable, but it is verifiable. Because of the environment, we cannot expect a policy specified to be completely upheld, because we cannot control the minds of users. What we can say, however, is that a stable solution to the problem of policy versus users will lead to a situation where the system remains in the acceptable regions of the lattice for most of the time (on average). A counterforce (police force, or immune system) can correct the minor transgressions which must inevitably occur.

But who says the policy will be stable, that transgressions will only be minor? It is, of course, possible to write system policies for a given mass of users which will provoke them into such rebellion that the policy will immediately fail. I claim that this is a good criterion for an unrealistic policy (governments sometimes make such mistakes) and that such a catastrophic failure is a pathology of the initial assumptions. The aim of system administration is never to build such unstable systems, so sufficient stability is just a basic requirement, a starting point.

This model of the user-machine interaction, constrained by policy, can be written in a more formal way, in order to map it onto well-known methods of stochastic dynamics. Suppose we examine any variable of the system, as a function of time. Suppose also that we collect the data over many periods (weeks) and examine the averages, calculated for all corresponding intervals. This provides us with an average picture of what the system is doing, in addition to an actual picture of what the system is doing. Now we define:

Actual value = average value + fluctuation

This split is significant for two reasons. The first is that the average value categorizes the approximate behavior of the system at any given moment, while the fluctuation tells us essentially about the variation of the environment. The second is that it separates microscopic from macroscopic, i.e., fast changes, or what happens over short times (fluctuation), from slow changes, or what happens over long times (changing average). We have thus formalized the idea that the environment is a complex changing signal which pokes and prods with much higher resolution than the stable part of the system.

In the lattice of changes, policy can only be associated with the stable part of the configuration. Acceptable levels of deviation from "perfect" can be used to define a distance from acceptable policy, or an average policy, but not an exactly enforceable one. The problem thus becomes, how can one keep the system as close as possible to an ideal policy-abiding state?

Can we curve the lattice, like a gravity well, so that the system rolls back into its point of lowest "energy," or most "policy correct" configuration (see Part 5)? This is the idea behind computer immunology. By building an immune system, or a mobilizable counterforce which regulates policy, one effectively builds such a gravity well. Unlike a gravity well, where all particles respond equally to the force, an immune system has a harder time of this job, because the lattice is multidimensional and the changes respond differently in each direction. This means that signatures and distinctions have to be made. Work of this kind has been done at the University of New Mexico using a method of classifying sequences of system calls inspired by the human immune system (see *http://www.cs.unm.edu/~immsec/*).

Let the Games Begin

In the future one can imagine feedback to users which indicates the state of the system. If users see a machine which is not "feeling well," this would be a signal to avoid that particular machine. This alone might be sufficient relief to allow the machine to correct itself (heal itself). This kind of bilateral feedback has been experimented with in artificial intelligence (e.g., the MIT Kismet robot; see *http://www.ai.mit.edu/projects/kismet*). I think it could be essential to the development of truly robust systems which interact with humans.

What happens when environment meets machines? The unpredictable meets the specified. If the machine is capable of adapting, there ensues a game of competition for the integrity of its design policy. If the machine cannot adapt, the specification ends up being ruined.

In a game theoretical model of system administration, it makes sense to divide users into those who obey policy and those who do not. Users who obey policy are irrelevant to the evaluation of policy because they can be absorbed into the background activity, i.e., the way in which the value of the "payoff" changes normally in time. On the other hand, if users do not obey policy, they might choose any number of strategies to try to confound it. A model of system administration is interested in evaluating how likely it is that such a strategy would succeed against policy.

Thus, at the simplest level, we think of the actors as motivated individuals who are in competition to maximize their gain or minimize their risk. They might work cooperatively, in an altruistic way, or non-cooperatively in a purely selfish way. There might be any number of players in a game, but the simplest case (also the first approximation) is to think of system behavior as a two-person game, in which the users of the system compete with the system itself for possession of valuables.

A game is characterized by a matrix (see Figure 2) in which the rows and columns are labeled by the strategies and counter-strategies of the players, and the body of the matrix contains the payoff to one of the players of interest. By using minimum/maximum techniques, one can seek the most effective mixture of strategies (represented by the histogram distributions), which leads to optimal results. In traditional games, the valuables of the game are easily identifiable game pieces or token rewards. In economics the reward is money; in natural sciences the reward is energy. In Part 6 of the series, I argued that rewards in a social setting are not only tangible assets, but can also be the vagaries of emotional reward: peer respect, personal satisfaction, aesthetics, and any number of others from our complicated emotional psyche. The relative importance of these pieces of the puzzle is also, in a sense, a matter of policy. Little is known in our field about the value-scales for the variability of human traits, but it would be surprising if such research had never been done

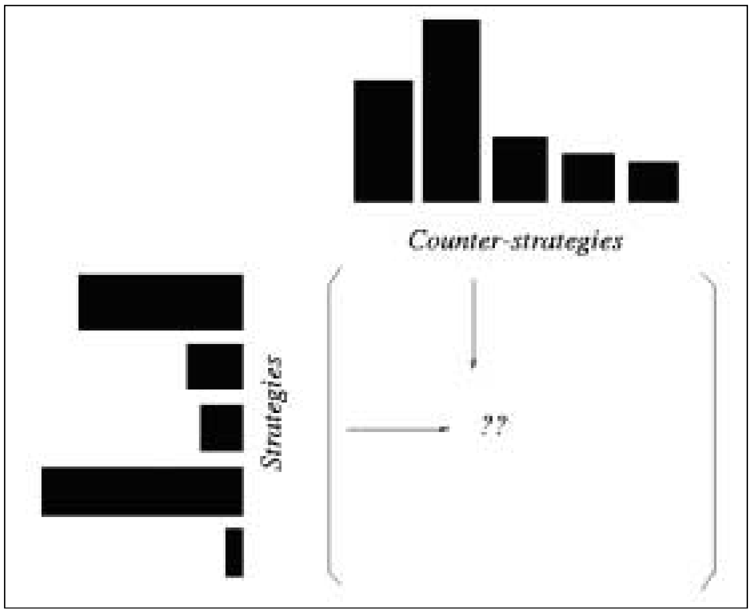

Figure 2

. . . system administration is neither a once-and-for-all solvable problem, nor a problem in which humans have to watch endlessly over their sheep

by psychologists in other contexts. With this viewpoint, there is a considerable simplification of the problem.

A game may be set up of hostile users versus a system policy agent (counterforce) and used to evaluate the optimal mixture of counterforce strategies, given that users do their worst. The game also doubles as a formal framework for finding out what the users' worst actually is. The solution of the game is one or more distributions and counter-distributions of strategies for each of the players, which the players can adopt in order to maximize their interest.

At the simplest level, one can assume that users do not cooperate with the system, but clearly one can extend the sophistication of the game in many ways to explore more refined possibilities. Evaluation of payoff is complex, and game playing is iterative. I can foresee that, in the future, simulation tools such as Petri Nets will play a role in simulating these complexities. It is not certain how much would be gained by this, but it is an avenue for further research.

Conclusion

In this series, I have tried to emphasize the dynamical, competitive aspect of complexity and the central importance of concentration (centralization) versus distribution: the management of entropy. Low entropy can be poison, high entropy dilapidation, but these are the extreme polarities of the scale. The issue is not a simple question of right or wrong; rather, it is one of seeking appropriate balance in the face of prevailing conditions. This theme recurs in many guises: Cray or workstation; central server or distributed database; uniqueness or redundancy; first-come, first-served (FCFS); or time-sharing, law-abiding users or disruptive users; knowledgeable users or ignorant users; automatic regulation or human intervention? These problems are all, at some level, about entropy management. The message, which applies in every case, is that the environment seeks out a balance between these strategies. We have the means to address these problems in quantitative terms.

I hope that I have drawn attention, in this series, to the idea that system administration is neither a once-and-for-all solvable problem, nor a problem in which humans have to watch endlessly over their sheep; rather, it is a process of continual regulation, a constant war against sickness, in which human or someday artificial ingenuity will occasionally be called upon to exceed the boundaries of simplistic programming. I have focused on what happens on computers, not on what happens in between computers (the network). The latter is another story altogether, far more complex, but building on what I have discussed here. At LISA this year, we will begin to see the results of our early probings into this challenging field.

writing the tk geometry managers in tcl

by Sergey Babkin

Sergey Babkin works for Caldera Systems on UnixWare/ OpenUNIX. He is also a FreeBSD developer and participates in a few smaller Open Source projects.

babkin@users.sourceforge.net

If you open a book on Tk programming, the internals of the geometry managers are usually considered a rather advanced topic. And the fact that there are not many geometry managers for Tk correlates well with this observation. A major obstacle in writing a geometry manager is that they are normally written in the C language and then linked with Tcl (or possibly Perl). To overcome this obstacle, I wrote a small module that implements the operations missing in Tcl/Tk but necessary for writing a geometry manager. This made writing the new geometry managers and experimentation with them much easier, so I want to share this experience.

Of course, an important question to consider is: why would someone want to write another geometry manager? Are not the managers provided with Tcl/Tk enough? Well, it depends on what do you want to do with them. For my project "Not A Commander" (see its home page at *http://nac.sourceforge.net*) they were not. I started this project in order to learn Tcl/Tk while doing something useful. And I had a quite good idea of what this useful thing would be: an X11 file manager done the way I like it.

I like to have the modal dialog windows shown within the main window of the application. When they are created as new top-level windows, one of two things usually happens: either a Netscape dialog pops up when I'm typing something in xterm, and my typing gets diverted to this dialog which immediately disappears; or the dialog gets lost among the other windows so that I am surprised when the main Netscape window refuses to react to any typing and mouse-clicking. Both cases annoy me greatly.

In a program I write for my pleasure, such dialogs should be shown in the main window, preferably as big as their natural size but not bigger than the size of the main window. Achieving this effect with the standard geometry managers is possible but far from easy, and even at its best does not work very well. For this reason, I decided to write my own geometry managers. And since I wanted to experiment with them easily, they had to be written in Tcl.

General Principles

The general principles of the geometry managers are described in the classic book *Tcl and the Tk Toolkit*, by John Ousterhout. The only caveat regarding this book is that it describes a quite old version of Tcl/Tk, so the details of the geometry managers' implementation have changed significantly. But the general principles are the same.

In short, they work as follows: as the slave widgets (or windows – for the purposes of this discussion these terms are synonyms) are configured, they calculate the size they need. They send these size requests to the geometry manager. The geometry manager collects these requests and places the slave widgets inside the master widget according to its policies and to the size of the master widget. If the size of the master widget is not enough to satisfy the requests of all the slaves, the geometry manager will usually resize some of the slaves to a smaller size than they requested.

The geometry manager also calculates the needed size of the master window that would fully satisfy the requests of its slaves and passes this request further up the hierarchy, to the master's master. Also, if the master widget gets resized, the geometry manager must recalculate the placement of the slave widgets in it accordingly. Because usually many slaves are created or changed at once, the geometry managers try to avoid the unnecessary work of doing the full recalculation on each and every change. They postpone the actual recalculation until the script becomes idle and then sweep up all the changes at once.

Writing the Geometry Manager

To write the geometry managers in Tcl I needed to implement in C the special operations not normally available to the Tcl scripts:

- Notify Tk that a slave widget will be managed by this manager.
- Notify Tk that a slave widget will no longer be managed by this manager.
- Map a slave widget within a master widget.
- Unmap a slave widget from a master widget.
- React to a geometry request from a slave.
- Send a geometry request further up.
- React to a reassignment of a slave to another geometry manager.
- Get the border width of the master window.

The last operation is theoretically not really necessary because the border width information usually can be obtained by running the `cget -bd` command of a widget, but in practice getting this information directly is safer and faster.

Two more operations are needed to connect the Tcl part of the geometry manager with the C services:

- Register a geometry manager with the C services.
- Unregister a geometry manager with the C services.

For reasons of space, I won't include the full listing of the C part; it can be downloaded (the file `geom.c`) as part of the Not A Commander (NAC) project. I will only describe the commands that are visible to the Tcl side and how they map to the Tk's C calls.

Since the code was written specifically for the NAC project, it doesn't try to be a proper module with, for example, namespace isolation. For the purposes of study, the simplified code seems to me more of an advantage than a drawback. Also it uses the NAC object model and conventions. For the examples in this article, I've gotten rid of most of these dependencies and dragged in only a minimal amount of them. The most important one is that the code is generally organized into "classes." The "classes" are not exactly what is meant by this word in the world of object-oriented programming but are a reasonable approximation. The procedures and global variables of a class are prefixed with the class name followed by a colon:

```
<class_name>:<object_name>
```

The object names that start with underscore are intended for the use of procedures of this class only (sort of like "private" and "protected" in C++ but without enforcement). More of these conventions will be described when we get to the Tcl code.

The commands implemented in C are:

Register a Tcl geometry manager:

```
nacgeom:register <manager_prefix>
```

`manager_prefix` is the class name of this manager. The implementation has an array of Tk per-manager structures, in which it finds a free slot and remembers the name.

Then it returns a manager ID which may be used for the subsequent calls. The most important part of the implementation is:

```
managers[freeid].name = strdup(argv[1]);
snprintf(interp->result, TCL_RESULT_SIZE, "%d", freeid);
```

UNREGISTER A TCL GEOMETRY MANAGER:

```
nacgeom:unregister <manager_prefix>
```

The implementation marks the structure in the registration array as free.

NOTIFY TK THAT THIS MANAGER WILL TAKE CARE OF THE SLAVES:

```
nacgeom:ofslave <mgr_id> <slave_window>...
```

mgr_id is the ID returned by nacgeom:register. More than one window can be specified. The most important call repeated for each slave window is:

```
Tk_ManageGeometry(win, &managers[mgr_id], (ClientData) mgr_id);
```

NOTIFY TK THAT THIS MANAGER RELEASES THE SLAVES:

```
nacgeom:freeslave <mgr_id> <slave window>...
```

The most important call in the implementation (repeated for each slave) is:

```
Tk_ManageGeometry(win, NULL, (ClientData) 0);
```

MAP A SLAVE WINDOW WITHIN A MASTER WINDOW:

```
nacgeom:map <slave_window> <master_window> <x> <y> <width>
<height>
```

x, y, width, and height are the position and size of the slave window in the master window. Tk provides a convenient function that takes care of all the necessary details:

```
Tk_MaintainGeometry(slave, master, x, y, width, height);
```

The Tk standard geometry managers separate a special case when the slave window is an immediate child of the master window; in this case they do all the mapping, positioning, and resizing by calling the low-level functions directly to improve the performance. However, the geometry managers written in Tcl are slow enough by themselves, so a little more overhead traded for convenience won't hurt them noticeably.

UNMAP A SLAVE WINDOW:

```
nacgeom:unmap <slave_window> <master_window>
```

This is implemented as another convenience call:

```
Tk_UnmaintainGeometry(slave, master);
```

REACT TO A GEOMETRY REQUEST TO A SLAVE.

This function passes the control in the opposite direction, it is called from Tk and calls a Tcl callback procedure that should be defined in the Tcl code:

```
<manager_prefix>:_geometry <slave_window> <width> <height> <border>
```

Width, height, and border width are the requested dimensions of the slave window. This function gets two arguments: the manager ID (as passed to Tk in Tk_ManageGeometry) and the slave window ID. Its important part is:

```
snprintf(bf, sizeof bf, "%s:_geometry %s %d %d %d", managers[id].name,
      Tk_PathName(win), Tk_ReqWidth(win), Tk_ReqHeight(win),
      Tk_InternalBorderWidth(win) );
Tcl_GlobalEval(my_interp, bf);
```

Send a geometry request up the hierarchy:

```
nacgeom:request <master_window> <width> <height>
```

This translates to the call

```
Tk_GeometryRequest(master, width, height);
```

React to a loss of slave due to its reassignment to another geometry manager.
Transfers the call to another Tcl callback function:

```
<manager_prefix>:_lost_slave <slave_window>
```

The implementation is similar to another callback:

```
snprintf(bf, sizeof bf, "%s:_lost_slave %s", managers[id].name,
      Tk_PathName(win));
Tcl_GlobalEval(my_interp, bf);
```

Get the border width of a window:

```
nacgeom:infobd <window>
```

This is implemented as

```
snprintf(interp->result, TCL_RESULT_SIZE, "%d",
      Tk_InternalBorderWidth(win));
```

The full text of the C support (geom.c) and the Makefile are parts of Not A Commander, and can be downloaded from *http://nac.sourceforge.net*.

An Example

Now let's look at an example of a full geometry manager that uses this interface: a simplified version of the post geometry manager from NAC. It allows posting of the slave widgets at the center of the master widget (if multiple slaves are posted, then they will overlap each other). The size of the slave widgets is limited only by the size of the master window.

The full text of the example is available for download from: *http://nac.sourceforge.net/pub/post.tcl*.

The procedures in the example do not follow the Tk convention of one command with many subcommands but instead follow the usual NAC naming conventions. The meaning of commands implemented in this manager is similar to the standard Tk geometry managers but simplified:

`post:add <slave-widget>... [-in <master-widget>]` – Manage the slave widgets
`post:forget <slave-widget>...` – Stop managing the slave widgets
`post:slaves <master-widget>` – Return the list of slaves posted in this master

To save space, the less essential parts are not shown here and are only briefly described. The script starts with loading the C part:

```
load ../geom.so nacgeom
```

The script expects that it would be placed in a subdirectory one level under the base directory of NAC.

Then three auxiliary procedures are defined. These procedures can be obtained by including the files gman.tcl and util.tcl from NAC but are defined in the script explicitly to avoid extra dependencies.

nacgeom:assert_ancestor <slave> <master> checks that the master and slave widgets conform to the proper ancestral relations, or throws an error otherwise.
bind_adduniqtag <window> <position> <tag> adds the binding tag to the binding list of the window at the specified position unless it's already on the list.
bind_rmclass <window> <tag> removes the tag from the binding list of the window (opposite of bind_adduniqtag).

The first action of the geometry manager itself is its registration with the C support code:

```
set post:gmid [nacgeom:register post]
```

The information about the widgets is stored in the global associative arrays indexed by the widget names. The value at the empty string ("") index is used to set the default values for the newly associated windows. For a master widget two variables are defined:

```
set post:slaves("") {}
set post:calcid("") {}
```

post:slaves contains the list of slaves posted in this master. post:calcid contains the delayed command ID of the scheduled geometry recalculation procedure. As I said before, when a Tk geometry manager gets a new slave or a geometry change request from a slave, it does not recalculate its geometry immediately because there is a good chance that more changes will follow immediately. Instead, it schedules its recalculation procedure for the time when the Tk process becomes otherwise idle.

For a slave widget the variables

```
set post:mymaster("") {}
set post:reqwidth("") 1
set post:reqheight("") 1
```

contain the name of its master and the size that it requested. This size can also be obtained by the Tk commands winfo reqwidth and winfo reqheight, but storing it in variables is more convenient. The procedure

```
proc post:_globals {} { ...
```

imports all the class global variables (those with names starting with post:) into the current function. When imported, these variables lose the class prefix in their names; for example, post:slaves simply becomes slaves. Normally, in NAC such a procedure (and a bit more) for a class would be generated automatically by calling

```
defclass post
```

But again to reduce dependencies in the example it's defined explicitly.

The binding tags are defined for the master and slave widgets:

```
bind post.master: <Destroy> {post:_forgetmaster %W}
bind post.master: <Configure> {post:_schedcalc %W}
bind post.slave: <Destroy> {post:forget %W}
```

When a master widget is destroyed, all of its slaves are freed. When a slave widget is destroyed, it's just forgotten as usual. When the master widget is resized by its own master, a geometry recalculation must be done for it. As always, this recalculation is not done immediately but scheduled for later.

The procedure that adds the slaves to a master is one of the two larger ones (another large procedure is for the geometry recalculation). Its arguments are like the Tk command place but with only one option supported.

```
proc post:add {args} {
    post:_globals
    set optpos [lsearch -regexp $args {^[^.].*}]
```

It starts with importing the class globals and finding where the options start in the argument list. All the widget names start with a dot, so anything not starting with a dot is considered an option.

```
    if {$optpos >= 0} {
        set opts [lrange $args $optpos end]
        set args [lreplace $args $optpos end]
    } else {
        set opts {}
    }
```

If any options are found, they are separated from the list of the new slave widgets. Since the packing order for this widget manager does not matter, the slaves are always added to the end of the list. And the only supported option is -in to select the master:

```
    set omaster {}
    foreach {opt val} $opts {
        switch — $opt {
        {-in} {set omaster $val}
        default { error "unknown post option $opt"}
        }
    }
    if {$args == ""} {
        # nothing to do
        return
    }
```

Having parsed the options, we add the slaves one by one:

```
    foreach win $args {
        if {$omaster == ""} {
            set master [winfo parent $win]
            if {$master == ""} {
                error "can't manage the root window as a slave"
            }
        } else {
            nacgeom:assert_ancestor $win $omaster
            set master $omaster
        }
```

If the master was specified explicitly, we need to assert that it's appropriate for this particular slave.

```
        if [info exists mymaster($win)] {
            post:forget $win
        }
```

If this slave is already posted, we need to forget it first. If the slave was previously managed by another geometry manager, Tk will notify that geometry manager automatically when we take over its slave. But if the slave was already managed by the same manager, then Tk will not send this notification to us, so we have to check for it.

```
            if { ![info exists slaves($master)] } {
                post:_initmaster $master
            }
            lappend slaves($master) $win
```

If this master has no slaves managed by this manager yet, we need to initialize our global variables and bindings for it. Then we add the new slave to its list.

```
            set mymaster($win) $master
            set reqwidth($win) [winfo reqwidth $win]
            set reqheight($win) [winfo reqheight $win]
            bind_adduniqtag $win end post.slave:
```

Then we initialize our global variables for the slave and add the post.slave binding to the end of its binding list.

```
            nacgeom:ofslave $gmid $win
```

We let Tk know that we take over the management of this slave.

```
            post:_schedcalc $master
        }
  }
```

Finally, for each posted slave we schedule the geometry recalculation procedure for its master. And that completes the adding of slaves.

The procedure for the opposite action, forgetting the slaves, is:

```
  proc post:forget {args} {
        post:_globals
        foreach win $args {
            if {![info exists mymaster($win)]} {
                continue
            }
            nacgeom:unmap $win $mymaster($win)
            nacgeom:freeslave $gmid $win
            post:_lost_slave $win
        }
  }
```

It does not take any options. Each slave is checked whether it's managed. The managed slaves are unmapped and Tk is notified that we don't manage them any more. Finally, we clean up our variables, and this cleanup happens to be the same as when Tk notifies us that the slave was moved by the user to another geometry manager, so we just call that procedure.

The last procedure of the user API returns the list of slaves for a master:

```
  proc post:slaves {master} {
        post:_globals
        return slaves($master)
  }
```

If no slaves are managed by this geometry mamager for this master it throws an error on an undefined variable.

The scheduling and unscheduling of the geometry recalculation is done with the following procedures:

```
proc post:_schedcalc {master} {
      post:_globals
      if {![info exists calcid($master)] || $calcid($master) == ""} {
          set calcid($master) [after idle "post:_recalc $master"]
      }
}
proc post:_unschedcalc {master} {
      post:_globals
      if {$calcid($master) != ""} {
          after cancel $calcid($master)
          set calcid($master) {}
      }
}
```

post:_schedcalc schedules the run only if it was not already scheduled, because one recalculation run is enough to process all the changes.

The geometry recalculation routine is the heart of a geometry manager:

```
proc post:_recalc {master} {
      post:_globals
      if {$slaves($master) == ""} {
          post:_forgetmaster $master
          return
      }
```

First we check that there are some slaves to manage. If no slaves are left, then this master does not need any more geometry management from us.

```
      set calcid($master) {}
```

Since the delayed command already has been called, its ID is not usable anymore, so we clean it. This is also a sign to post:_schedcalc that when called it must schedule a new delayed execution of the geometry recalculation.

```
      set bd [nacgeom:infobd $master]
      set maxwd [expr [winfo width $master] - $bd *2]
      set maxht [expr [winfo height $master] - $bd *2]
```

The geometry manager should respect the internal border of the master window and not use it for placing the slaves. For this simple manager this just means that the border width must be deducted from the available size.

For geometry managers that do not change the size of the master, such as this post or the standard place, we can now start mapping the slaves. However, the geometry managers that propagate the geometry requests up the widget hierarchy should first calculate the size of the master necessary to accommodate all its slaves as requested and pass this request up. In a hypothetical case of a geometry manager that tries to make the master widget as big in each dimension as the largest size requested by a slave, this code might be:

```
set reqwd 1
set reqht 1
foreach win $slaves($master) {
      if {$reqwidth($win) > $reqwd} {
          set reqwd $reqwidth($win)
      }
      if {$reqheight($win) > $reqht} {
          set reqht $reqheight($win)
      }
}
if {[winfo reqwidth $master] != $reqwd
|| [winfo reqheight $master] != $reqht} {
      nacgeom:request $master $reqwd $reqht
      post:_schedcalc $master
      return
}
```

The smallest valid size for a widget is 1. So the calculation starts with this value. If some slave has requested a larger size, we take this larger size. After processing all the slaves we check whether our new calculated size is different from the size we calculated last time (and passed further up). If it's the same, we can start mapping the slaves. If it has changed, the request for the new size is passed to Tk, which passes it to the geometry manager that has our master as a slave. That geometry manager will schedule its own geometry recalculation for later. There is a good chance that by results of this recalculation it will change the size allocated to our master according to our request.

So for now, mapping the slaves based on the old size of the master widget would be a waste of time, and the best thing we can do is to schedule our own recalculation for later and return. If all goes well, the upper geometry manager's scheduled recalculation will run first (because presumably it was scheduled first) and set the new size to our master. Then our recalculation will run again and map the slaves according to the new size. However, if our master's master wants to resize itself as well, then our rescheduled procedure would run before the resizing happens at the upper level. But it's not a big problem: the only loss is time spent on an extra run of recalculation. Then when the upper geometry manager finally resizes our master widget, it will cause a configure event in this widget which we have bound to the recalculation request, so eventually our recalculation will run again and redo everything based on the new available size.

Now let's return from that hypothetical case to the post geometry manager. We map each slave in its turn. First we calculate its dimensions (wd and ht) and position (atx and aty) and then we actually map it:

```
      foreach win $slaves($master) {
          if {$maxwd <= 0 || $maxht <=0} {
              nacgeom:unmap $win $master
              continue
```

Zero or negative maximal dimensions of the slave may occur if the size of the master widget is less than its border width. Since there is no way to display the slave, we unmap it.

```
          } else {
              set wd $reqwidth($win)
              if {$wd > $maxwd} {
                set wd $maxwd
            }
            set ht $reqheight($win)
            if {$ht > $maxht} {
                set ht $maxht
            }
            set atx [expr ($maxwd-$wd)/2]
            set aty [expr ($maxht-$ht)/2]
        }
```

The size of the slave in each dimension is limited by the available space. Then the slave's position is centered.

```
        incr atx $bd; incr aty $bd
        nacgeom:map $win $master $atx $aty $wd $ht
    }
}
```

Finally, the position is adjusted for the border width, and the slave is mapped at its calculated position.

When the first slave is added to the master, the following procedure is called to initialize the master's data structures:

```
proc post:_initmaster {master} {
    post:_globals
    set slaves($master) {}
    set calcid($master) {}
    bind_adduniqtag $master end post.master:
}
```

It also adds a bind tag which allows us to react to the master widget's destruction or resizing by the master's master.

When the master widget is destroyed or loses its last slave, its data should be cleaned up. All this cleanup activity is done in the next procedure:

```
proc post:_forgetmaster {master} {
    post:_globals
    if [info exists slaves($master)] {
```

If there is no data for this master then there is nothing to clean up. This check also gives some additional safety against double calling of this function due to some race condition.

```
        eval "post:forget $slaves($master)"
```

Any slaves that are left over should be forgotten. If the slaves list is empty, `post:forget` will just do nothing.

```
        post:_unschedcalc $master
```

If the recalculation was scheduled, it must be canceled. Otherwise when it runs later it would find no data entries and throw an error. This cancellation can be done only after forgetting the slaves because when a slave is forgotten, the master's geometry recalculation gets scheduled.

```
            bind_rmclass $master post.master:
            unset slaves($master)
            unset calcid($master)
        }
    }
```

Finally, we remove the binding tag and free the per-master data entries.

When we stop managing a slave, a cleanup should be done as well. This is handled by the procedure post:_lost_slave, which is called in the following cases: a slave is forgotten on a user's call to post:forget; a slave is destroyed and post:forget is called through the binding of the destroy event; a slave is passed to another geometry manager by the user and this procedure is called as a callback from the supporting C code.

```
proc post:_lost_slave {win} {
        post:_globals
        if {![info exists mymaster($win)]} {
            return
        }
```

As with the masters, we'd rather be safe than sorry and not try to free data that is not allocated.

```
        set master $mymaster($win)
        set idx [lsearch -exact $slaves($master) $win]
        if {$idx >= 0} {
            set slaves($master) [lreplace $slaves($master) $idx $idx]
            post:_schedcalc $master
        }
```

This slave is removed from its master's list, and the master's geometry recalculation is scheduled. The recalculation is not absolutely necessary for this particular geometry manager because the slaves are posted independently of each other. But in a generic case, forgetting a slave may cause serious changes in the master's geometry. Even for this manager, however, doing a recalculation is a good thing; if this were the last slave of this master, the geometry recalculation will catch it and free the master's data structures as well. Otherwise we would have to check here for this case explicitly.

```
        bind_rmclass $win post.slave:
        unset mymaster($win)
        unset reqwidth($win)
        unset reqheight($win)
}
```

Finally, we remove the binding tag and free the per-slave data entries.

When a slave sends a new geometry request, the C portion of the code forwards the call to the callback procedure:

```
proc post:_geometry {win wd ht bd} {
        post:_globals
        if [info exists mymaster($win)] {
            set reqwidth($win) $wd
            set reqheight($win) $ht
            post:_schedcalc $mymaster($win)
        }
}
```

If the slave is associated with a master, we remember the values it requested and schedule the geometry recalculation. Otherwise we consider this a spurious call and do nothing.

This completes the simplified post geometry manager. The post geometry manager in NAC has many more features, such as margins around dialog windows and completely different logic for the posting of menus.

Customized geometry managers open many other interesting possibilities. Some of the more complex examples that may be found in Not A Commander include:

- An auto-wrapping label widget (see the classes awlabel and awlpack in wdgt.tcl). If the label can not get enough space along the X axis, it wraps the text at the available width and tries to extend itself vertically. This is achieved by composing the widget from the Tk label subwidgets and controlling them with a highly specialized geometry manager. Of course, this effect may be achieved much more efficiently by modifying the implementation of the Tk label widget, but the internals of that widget are far from simple and are difficult to modify.
- A scrollbar displayed automatically when there is not enough space for all the slaves (see the class menupack in gman.tcl). Note that this is different from the Perl/Tk widget "Scrolled" in which the scrollbars are displayed all the time. The customized geometry managers allow the scrollbars to be displayed only when they are really necessary – that is, when there is not enough space for all the slaves.
- A pseudo-grid (see the class pgrid in gman.tcl). It implements a two-level composition: the whole grid consists of a set of row widgets, each of the rows containing a few slave widgets. The rows are combined by some other geometry manager (such as Tk's pack). The pseudo-grid manager controls the slaves within the rows so that the columns within each row are aligned with each other row. The pseudo-grid is extremely convenient for the vertical menus: each row is a composite menu button widget while the slaves are the items within these buttons. These items are arranged into non-overlapping columns: the optional radio/checkbutton indicator, the button label, and the accelerator key label.

Of course, the downside of implementing the geometry managers in Tk is that they are quite slow compared to those implemented in C. Because of this they do not scale well to a large number of managed widgets and work best either for special cases involving a small number of widgets or for prototyping with a following rewrite in C.

the tclsh spot

by Clif Flynt

Clif Flynt is president of Noumena Corp., which offers training and consulting services for Tcl/Tk and Internet applications. He is the author of *Tcl/Tk for Real Programmers* and the *TclTutor* instruction package. He has been programming computers since 1970 and a Tcl advocate since 1994.

clif@cflynt.com

The previous *Tclsh Spot* article described a simple telnet client that would report the initial configuration options. This article will expand the sniffer into a client that can interact with a server, maintain its internal state, and respond to various commands the server can send. In the course of this, I'll demonstrate some things we can do with the Tcl namespace.

The Tcl namespace command provides a private, named area in a Tcl script where data and procedures can exist without interfering with other data and procedures that might have the same names. A Tcl namespace can implement most of the capabilities of a Java or C++ class.

A Tcl script is most useful when it's merged into other Tcl code. You can merge one Tcl script into another with the source command which loads a script into a Tcl program, and evaluates the commands in that script before evaluating the next line of the original script.

Syntax: source *fileName*

This is similar to the C language #include or the C-shell source command. This is a simple technique, and it works well for many applications.

However, if you source two packages that have overlapping names for variables or procedures, the last package you load will overwrite the procedure body or data values set by the first package.

We can use the Tcl namespace command to create a private, named area in our telnet client where data and procedures can exist without interfering with other data and procedures that might have the same names.

A namespace is created with the namespace eval command:

Syntax: namespace eval *namespaceID arg ?args?*

namespace eval	Create a namespace, and evaluate the script arg in that scope. If more than one arg is present, the arguments are concatenated together into a single command to be evaluated.
namespaceID	The identifying name for this namespace.
arg ?args?	The script or scripts to evaluate within namespace namespaceID.

The variables defined within a namespace will last until the namespace is destroyed by the namespace delete command. This makes a namespace an ideal place to keep a package's internal state.

The state information for the telnet sniffer application was held in a global associative array. A telnet namespace with that array included in it can be created with code like this:

```
namespace eval telnet {
    variable Telnet
```

The variable command declares a Tcl variable within a namespace. When the variable command is used outside a procedure, it creates a variable within a namespace and initializes it to an optional value. When the variable command is used within a procedure, it maps a variable from that namespace scope into the procedure's local scope.

Syntax: variable *varName ?value? ?var2? ?val2?*

varName	The name of the variable to create, or map into local scope.
value	An optional value for this variable. If the variable already has a value, the new value will overwrite the old.

The sniffer program used two global arrays, the telnet array that held the state information, and the Lookups array that was used to map from hex values to human-friendly information strings. The Lookups array was created at run time by scanning the telnet.h include file and massaging the #define xx yy lines into Tcl array assignments in a simple procedure.

Any Tcl code can be evaluated within the namespace eval body. We can evaluate the readIncludeFile procedure to create the Lookups array within the telnet namespace, and can even use the source command to load the script that includes this procedure into the namespace.

```
namespace eval telnet {
      variable Telnet

      source readincl.tcl

      readIncludeFile Lookups /usr/include/arpa/telnet.h
      set Lookups(UNKNOWN.-1) "Unknown Option"
```

By sourcing the readincl.tcl script within the namespace, the readIncludeFile procedure is defined within the namespace. This procedure is available for use within the namespace but is not easily visible from the outside world.

Tcl namespaces are a tree-structured construct, like a file system or graphics windows. Where a POSIX-style file system uses a slash to separate parent from child directory, namespaces use a double colon to delimit parent and child namespaces.

When Tcl is started, the default, top level, namespace is ::. The namespace eval telnet {...} command creates a new namespace ::telnet.

Unlike Java or C++ classes, there is no privacy in a Tcl namespace. The Tcl philosophy is to make as much information as possible available to the programmer. Thus, you can always access a member of a Tcl namespace by its full namespace pathname.

We can define a procedure within the telnet namespace with a normal looking Tcl command like:

```
namespace eval telnet {
      ...
      proc openSocket {address} {
          variable Telnet
          set Telnet(socket) [socket $address 23]
      }
}
```

We could invoke the procedure from outside the namespace with a command like:

```
::telnet::openSocket 127.0.0.1
```

Java and C++ let the application programmer know the private and public API by restricting access to non-public methods. Since any procedure within a namespace can be invoked from outside the namespace, we need a mechanism to let the programmer know which are public and which are private methods.

One convention used within Tcl is that public methods start with a lowercase letter and private methods start with an uppercase letter. The rationale is that it takes an extra keystroke to type an uppercase letter. This forces programmers to recognize that they are violating the package's intended use by calling this procedure.

The Tcl namespace also includes a namespace export command to declare which procedures are part of the external API.

Including this line in the namespace eval body tells the application programmer that the external API for this namespace is the openSocket, binarySend, and telnetEvent procedures.

```
namespace export openSocket binarySend telnetEvent
```

Along with using a namespace to hide variables, we sometimes use a namespace to hold just procedures. This ensures that we don't have procedure-naming collisions when our application sources several files.

We could define a namespace that has functions to convert strings of binary data to hex digits like this:

```
namespace eval binaryStrings {
    namespace export bin2hex hex2bin

    proc bin2hex {binaryString} {
        binary scan $binaryString "H*" hexData
        regsub -all {..} $hexData {\0 } hexList
        return $hexList
    }

    proc hex2bin {string} {
        regsub -all " " $string "" string
        set line [binary format "H*" $string]
        return $line
    }
}
```

One of the strengths of the Tcl namespace is that they can be nested. This implements the OO composition (or "has-a") feature. For example, the binaryStrings namespace can be embedded in the telnet namespace with code like:

```
namespace eval telnet {
    source binaryStrings.tcl
    ...
}
```

Just as file systems support absolute naming by starting from the root file system (/usr/local/bin/tclsh) or relative naming, by starting from the current directory (subdir/file), Tcl namespaces can be accessed by either absolute or relative names.

The procedures in binaryStrings can be invoked from within the telnet namespace as: ::telnet::binaryStrings::hex2bin or binaryStrings::hex2bin.

Since the telnet namespace may be included in another namespace, it's best to use relative naming when embedding one namespace within another.

Alternatively, since the public API for the binaryStrings namespace is exported, we can use the namespace import command to import those procedures into the current namespace. This would let us invoke the procedures without any namespace prefix.

```
namespace eval telnet {
      source binaryStrings.tcl
      namespace import binaryStrings::*
      ...
      set hex [bin2hex $binaryValue]
}
```

Using the namespace import command undoes the protection from procedure-name collisions that we got from using namespaces, but within a controlled environment (like a namespace), the namespace import command can make code easier to read.

These techniques let us set up a telnet namespace for the sniffer that was developed in the last article. In order to handle real telnet client-server interactions, the script needs to be able to handle the negotiation commands and retain state information about the supported and unsupported options.

The telnet protocol includes a lot of subnegotiation options ranging from a need to echo characters to supporting defining data rates and terminal size.

Implementing these options follows a pattern – they have a current value and react to demands that the option be supported or not supported.

The reaction to a command varies depending on the state of that option. For example, if an option's status is to be supported, and the server sends a message to not use that option, the client should send a return message that the option won't be used. However, if the option was already turned off, no reply should be sent.

We could implement this with a set of objects that include the current state information and procedures to report the contents. We could use one object for each option, and use Tcl's ability to nest namespaces to hold all the option namespaces with the telnet namespace.

If we were writing this in a true object-oriented language like C++ or Java, we might have a base class for options and two derived classes for supported and unsupported options.

We can implement the inheritance (or "is-a") feature of true OO languages with the namespace command by using a base command to initialize a namespace and another set of commands to set the personality of the class.

In C++ terms, the script used to initialize the namespace is the base class, and the personality commands implement the methods in the derived class.

We can use the hex values of the commands to name the procedures in the option namespaces, just as we did in the sniffer program. This makes the parsing simpler (let Tcl do it). Because we have separate namespaces for the various procedures (fb, fd, 01, etc.), they don't conflict with each other or with the procedures sharing that name in the telnet namespace.

We can define the names of the namespaces on the fly. To make the code easier to write (if a bit more cryptic for reading), I'm using a naming convention of XX.TELOPT for the option namespaces, where XX is the hex value of the option number.

The code to define namespace objects for unsupported features looks like this:

```
set baseClass {variable clientValue %s supported %s;}

foreach cant {00 05 21 22 23 24 25 26 27 } {
      namespace eval $cant.TELOPT [format $baseClass "" 0]
      namespace eval $cant.TELOPT [format "proc fb {} {return fffe%s}" $cant]
      namespace eval $cant.TELOPT [format "proc fd {} {return fffc%s}" $cant]
      namespace eval $cant.TELOPT  "proc 01 {} {return {}}"
      namespace eval $cant.TELOPT [format \
          {proc fe {} {variable supported;
              if {$supported} {
                  return fffc%s
              } else {
                  set supported 0;
                  return {}}
          }} $cant]
}
```

For supported options, it's a bit more complex, since some options have data that must be reported when requested. Here's code that will create objects for the supported options.

```
foreach {can initialVal}   {18 "dumb"
                            1f "0x00500018"
                            20 "57000,57000"
                            03 "01"
                            01 "01"} {
      namespace eval $can.TELOPT [format $baseClass $initialVal 1]
      namespace eval $can.TELOPT [format "proc fd {} {return fffb%s}" $can]
      namespace eval $can.TELOPT [format "proc fb {} {return fffd%s}" $can]

      if {[string first 0x $initialVal] == 0} {
          set hex [string range $initialVal 2 end]
      } else {
          binary scan $initialVal H* hex
      }

      proc $can.TELOPT::01 {dummy} [format "return fffa%s00%sfff0" $can $hex]

      namespace eval $can.TELOPT {proc 00 {val} {variable serverValue;
          set serverValue $val}}

      namespace eval $can.TELOPT [format \
          {proc fe {} {variable supported;
              if {$supported} {
                  return fffc%s
              } else {
                  set supported 0;
                  return {}}
          }} $can]
}
```

The code to support the fb, fc, fd, and fe commands (WILL, WON'T, DO, and DON'T) in the telnet namespace are fairly simple. They look like this:

```
# WILL command fb : Reply DO(fd) or DONT(fe)
proc fb {subl textName oobName} {
      variable Telnet
      upvar $oobName oob
```

```
        set opt [lindex $subl 0]
        AddInfo $subl TELOPT_
        append oob [eval $opt.TELOPT::fb]

        return 1
  }
```

The fa (SUBNEGOTIATION) command is a bit trickier, since the negotiation may be a request for the value, or a value being supplied.

The format for this command is either

```
ff (IAC) fa (SB) 1 (request data) ff (IAC) f0 (SE)
```

or

```
ff (IAC) fa (SB) 0 (provide data) DATAVALUES ff (IAC) f0 (SE).
```

The code to implement the fa command is:

```
  proc fa {subl text oobName} {
        variable Telnet
        upvar $oobName oob
        # Starts with character after the 'fa'
        Debugputs "Dealing with: [lrange $subl 0 20]"
        set count 1
        set type [lindex $subl 0]
        set action [lindex $subl 1]
        foreach {p2 data} [FindFFF0 $subl] {}

        append oob [$type.TELOPT::$action $data]
        AddInfo $subl TELOPT_
        return $p2
  }
```

That covers the interesting parts of this script. As usual, the full source code (about 450 lines) is available at *http://noucorp.com/cgi-bin/noucorp/generic.tcl?dir=/home/httpd/html/tcl/login.*

variable length arrays

We've been looking at some of the features added to C9X, the recent standards update to C. In this column we'll consider the use of variable length arrays (VLAs).

by Glen McCluskey

Glen McCluskey is a consultant with 20 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documen-ta-tion areas.

glenm@glenmccl.com

Some Basics

Suppose that you need to allocate an array in your program, but when you're writing the program, you don't know how long the array should be. What do you do in such a case? An obvious answer is to use malloc() and dynamic allocation. This approach will certainly work, but has a couple of problems. One is that you need to worry about freeing up the storage when you're done with it to avoid memory leaks, and another is that dynamic allocation for multidimensional arrays gets complicated.

C9X offers another approach, the use of VLAs. Here's an example of what such usage looks like:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    if (argc != 2) {
        fprintf(stderr, "Missing numeric argument\n");
        return 1;
    }

    int n = atoi(argv[1]);

    int x[n];

    printf("sizeof = %d\n", sizeof(x));

    return 0;
}
```

A numeric value representing an array length is passed to the program, and an array of this length is allocated. When the array goes out of scope, its storage is automatically reclaimed.

The size of the array is calculated at run time. For example, if you specify an argument of 10, and the size of an int on your machine is 4, then 40 will be printed.

The array is of fixed size once it's allocated, but its size is not fixed until the flow of control passes the declaration.

Variable Length Arrays as Function Arguments

Here's another example of how you can use VLAs, passing them as function arguments:

```
#include <stdio.h>

typedef void (*fp)(int, int[*][*]);

void f(int, int[*][*]);

int main()
{

    int n = 2;
```

```
        int x[n][n];

        x[0][0] = 1;
        x[0][1] = 2;
        x[1][0] = 3;
        x[1][1] = 4;

        fp fptr = &f;

        (*fptr)(n, x);
        fptr(n, x);

        return 0;
}

void f(int n, int x[n][n])
{
        printf("0,0 = %d\n", x[0][0]);
        printf("0,1 = %d\n", x[0][1]);
        printf("1,0 = %d\n", x[1][0]);
        printf("1,1 = %d\n", x[1][1]);
}
```

In this example a 2 x 2 VLA is created and then passed as an argument to a function. The called function is declared before use, along with a function pointer typedef. Note how the [*] notation is used to specify VLA parameters.

Pointer Arithmetic

In the first example above, we showed how sizeof() returns a dynamic value, known only at run time. This same consideration also applies to other calculations, such as pointer arithmetic. Consider the following example:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
        if (argc!=2) {
            fprintf(stderr, "Missing numeric argument\n");
            return 1;
        }
        int n = atoi(argv[1]);
        int arr[10][n];
        int (*p)[n] = arr;
        arr[4][n-1] = 37;

        p += 4;

        printf("%d\n", p[0][n-1]);

        return 0;
}
```

The VLA arr becomes a 10 x n array, with n set at run time. We initialize a pointer to the array, store a value at [4][n-1] in the array, and then increment the pointer by 4. In this situation, saying p += 4 means that four rows of the array should be skipped, but the length of a row (the number of columns) is not known to the compiler and must be dynamically evaluated.

The variable p in this example uses what is known as a "variably modified type." The line

```
int (*p)[n] = arr;
```

declares p to be of variably modified type and then initializes it with arr. p is a pointer to an array of n integers, and the initialization sets p to point at the VLA. [n] is part of the variably modified type.

VLAs are a subset of variably modified types. Such types must be declared at block or function prototype scope. So, in this example:

```
int n = 5;
//int (*p)[n];
void f()
{
      int x[n][n];
      int (*p)[n] = x;
}
```

uncommenting the global declaration will trigger a compile error.

Restrictions on Variable Length Arrays

There are some things you can't do with VLAs. One of them is to use {} initializers, like this:

```
void f(int n)
{
      int x[n] = {1, 2, 3};   /* can't do this */
}
```

One problem with allowing this usage is that the value of n is not known to the compiler, so it's impossible to determine whether too many initializer values have been specified.

Another thing you can't do is to allocate a VLA using global or static storage:

```
int n = 3;
int x[n];
void f()
{
      static int y[n];    /* can't do this */
}
```

There's no way at compile time to determine how big these arrays will be.

A third example concerns the use of sizeof, like this:

```
void f()
{
      int n = 3;
      int x[n];
      int y = 5;

      switch (y) {
          case sizeof(n):
              break;
          case sizeof(x):  /* can't do this */
              break;
```

```
        }
}
```

The usage in the second case label is invalid because the size of x is not known to the compiler.

Finally, it's illegal to jump around the declaration of a variable length array:

```
void f()
{
        int n = 3;
        int y;
        goto lab;       /* can't jump around decl below */

        int x[n];

lab:

        y = x[0];
}
```

Static and Restrict

There's another interesting aspect of VLAs that ties in with performance and optimization. When you're specifying variable array parameters to a function, you can use the static and restrict keywords:

```
#include <stdio.h>

double f(int n, double x[static n])
{
        double sum = 0.0;

        for (int i = 0; i < n; i++)
            sum += x[i];

        return sum;
}

int main()
{
        int n = 10;
        double x[n];
        for (int i = 0; i < n; i++)
            x[i] = (double)(i + 1);

        double sum = f(n, x);

        printf("%g\n", sum);

        return 0;
}
```

Using static tells the compiler that the underlying pointer used to hold the VLA argument (1) is not NULL, (2) points to elements of double type, and (3) points to at least n elements which are guaranteed to be available.

This information can be used to initiate loads or prefetches of the arrays that are accessed within the function. Another example uses both static and restrict:

```
#include <stdio.h>
```

```
void f(int n, double x[static restrict n],
      double y[static restrict n])
{
      for (int i = 0; i < n; i++)
          x[i] += y[i];
}

int main()
{
      int n = 10;
      double x[n];
      double y[n];
      for (int i = 0; i < n; i++) {
          x[i] = (double)(i + 1);
          y[i] = (double)(i + 100);
      }

      f(n, x, y);

      for (int i = 0; i < n; i++)
          printf("%d %g\n", i, x[i]);

      return 0;
}
```

In this example, the array parameters to f() are guaranteed to be (1) non-NULL, (2) of type double, (3) at least of length n, and (4) unique and non-overlapping. Such information can be used to generate optimized code.

Variable length arrays are especially useful in numerical programming, and also in situations where you don't know the array size at compile time, and you don't want to deal with all the complications of dynamic allocation.

musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V*.

rik@spirit.com

I just got accused of being anti-open source. Because I had mentioned the BIND weakness being exploited by the Lion worm, I was suddenly the enemy. I know that I sometimes suffer from what I call "executive reading" – the ability to read plain text and totally misunderstand important sections of it. I invented this term in jest based on the responses I often get when exchanging email with busy people. I thought this was a disease that comes with age.

The executive-read seems to happen to me just when I am already on the edge of exploding, and I read something that appears to be the most dubious thing ever. Sometimes it is. And sometimes it makes better sense when I slow down enough to really comprehend what the author was saying.

I'd like to set aside the notion that I am against open source. I consider open source, including most of its variants, a wonderful idea. And, with that out of the way, I'd like to rave for a few paragraphs.

Linux systems have been the most commonly exploited UNIX platforms for many years now. And why is this? Is it because the open source community doesn't examine its own code? Or perhaps because the code wasn't examined before it was released, so that it could be exploited later? Maybe it is because the programmers are working at Internet speeds, and a couple of little problems slipped past them.

The real issue with security problems in Linux has little to do with any of these things. Certainly, better code review would help. The OpenBSD folk have made a serious effort at this, and Web defacement statistic sites like attrition.org and alldas.de reflect this, even though there are a LOT more Linux Web servers in the world than BSD-based ones.

The security problems we face today go beyond code review, however. What we face instead is a design crisis.

Out-of-the-Box

Just take any recent version of Linux and do a vanilla workstation install. And you know what? You have just become a server, and you might not even know it. You will have over a dozen listening TCP servers, so not only are you ready to rock and roll but you also have just opened your system up for potential attacks on all of these ports. And why?

Ease of use.

You can now perform DNS lookups without having to enter an IP address in /etc/resolve.conf. You can have email sent directly to your system (if an MX record already exists for your IP address, that is). The finger daemon is ready to reveal your login name, and the r commands are just waiting to serve. At least Linux doesn't come with an /etc/hosts file with a lonely plus sign in it, welcoming all who might drop by, the way Sun Microsystems did for so many years.

One of the simplest things that anyone can do to reduce the threat of network attacks is to disable unnecessary network services. Shut off all the services, and your only network footprint becomes the knee-jerk responsiveness of the IP stack to certain ICMP messages and broadcasts.

But, from the perspective of a network-based attacker, you have just become invulnerable. Nothing they can send you on the network will give them a shell prompt, execute a command, delete a file, or divulge any information other than the identity of your OS. Perhaps an attacker can convince you to do something dumb – social engineering is a powerful mechanism, as old as fraud – but without your unwitting assistance, you have configured the rock of Gibraltar.

If it is really that simple, why don't more people do this? Even better, why don't vendors do this for them? The answer is that vendors know that operating systems preconfigured to "just work" sell better than those that take a bit of tinkering to get them to do anything. And you can buy or download operating systems that are set up correctly already. There are Linux distributions configured out-of-the-box for better security. And, of course, there's OpenBSD.

While you can make a UNIX system invulnerable to network attacks pretty easily, the same is definitely untrue about MS systems

Feature Quest

The quest for ever more interesting features has a much more enthusiastic participant than any open source group. I speak of Microsoft, which I will denote as MS.

While you can make a UNIX system invulnerable to network attacks pretty easily, the same is definitely untrue about MS systems. One of the saddest things I have to say when teaching security classes is that the easiest way to attack an MS box is with email. MS, in its quest for features, first loaded its browser, Internet Explorer, up to its gunwales with features, making the remote execution of code on a targeted machine child's play. Or should I say script-kiddie play?

Then, by linking IE to Outlook and Outlook Express, you can now send email and expect to have interesting things happen. Note that if you plan on updating every MS platform in your organization every three months or so (or whenever the next gaping hole is uncovered), you should be okay. Of course, everybody already does this, right? A better way of looking at this problem is to realize that if you are using a year-old version of IE, you should expect to have a Trojan installed on the system every couple of weeks. Fortunately, you do have anti-virus software installed to detect the Trojans that have now become endemic on MS platforms, don't you?

Once upon a time, MS platforms weren't considered interesting enough to bother hacking. Now, when an $800 PC comes with an 800MHz processor, scads of disk space, and may be connected full-time via cable modem or DSL, MS boxes have become a lot more interesting. Last time I checked, you could download over 100 different variations of MS Trojans (the source code, I mean, so you can create your own variation). A popular twist is to use private IRC channels for remote control. And the people controlling these channels typically instruct the Trojan to upload a new version every several days or so.

Wouldn't want to have an out-of-date remote control Trojan running on victims' systems.

MS XP

Speaking of remote control, MS has promised to "fix" the consumer marketplace. With MS XP, Windows NT finally comes to the consumer desktop, with an announced release date of October 25. Systems appearing in stores will no longer come with insecure Windows 98, but instead with a security-enhanced Windows XP Home Edition. Let's check out some of the features.

. . . the reality of it is that it only takes a tiny group of people to take remote control of Internet connected systems designed with flexibility and features instead of reasonable security

You can get an idea of what is in store for MS users by visiting *http://www.microsoft.com/windowsxp/home/guide/dependable.asp*. Let's try a short quote:

"Using Remote Assistance, you can turn over control of your computer to a friend or technician who can solve your technical problems – without visiting your home. Once you give permission, the other person can control your computer remotely, over a network. . . . For extra security, you can also set a password that the recipient must use to connect to your computer."

I like that. It is as if MS has built BO2K or SubSeven right into Windows XP. I wonder if you can control the CD drawer too? And you can even set a password. I guess this is what MS meant by "enhanced security." Let's look at the next feature.

"Network Setup Wizard makes it easier than ever to set up your own home network so you can share printers, devices, files, and Internet connections among all the computers in your home." Sounds like a great idea, making all of that unused disk space available for remote use.

"System Restore: If you experience system failure or another significant problem, you can use the System Restore feature to roll back your computer to a previous state when it was working normally." Now this sounds really useful. The next time NTFS corrupts a critical file, or installing a game overwrites a key .DLL with an older version, you might actually recover without re-installing. This one feature alone sounds like a good reason to upgrade, as it will pay for itself in days.

There is even an "Internet firewall" included. Too bad it won't block email attachments, VB Script, HTML, XML, JavaScript, and the dozen other things that have proven dangerous for MS systems.

Lost

It is as if somehow the designers of operating systems got lost along the journey to the future. They believed it is all fun and games, and hey, we trust everybody! When the reality of it is that it only takes a tiny group of people to take remote control of Internet connected systems designed with flexibility and features instead of reasonable security.

Operating systems can be designed like ships, and I don't mean the Titanic. The notion of sandboxes, similar to the watertight compartments in Navy ships, would be a great addition to operating systems, especially if it included real hardware support to facilitate a strong implementation. Instead, we have the Titanic, a poorly designed ship, but boy, was it fast, and the accommodations (at least on the upper decks) were wonderful.

Back in 1982, when microprocessors were getting really cheap, I thought I saw the writing on the wall. I was working as a consultant at Morrow Designs, and they were designing disk controllers, and even serial port cards, with their own embedded processors. You build a system with distributed intelligence instead of having a single processor that has access to the entire system.

Let's try an analogy from Star Trek (whatever generation). There is always a method for self-destruct, to prevent the starship from falling into enemy hands. Enabling this self-destruct mechanism takes key phrases provided by the command staff (and always at least two members). Good security design, and quite appropriate given the seriousness of the occasion.

Now, if the Enterprise were designed like a modern operating system and its underlying hardware, self-destruct buttons would be sticking out of the walls, hanging from the ceiling, located next to the "Flush" button, and of course, right where your hand would reach to turn on the light in the middle of the night.

Our efforts to secure our existing systems resemble the crew of this sorry Enterprise going around putting big warning signs next to all the self-destruct buttons, as well as taping plastic cups over the ones little kids might press just for the heck of it (there are always a few kids on Federation warships it seems).

WARNING!! DO NOT PRESS THIS BUTTON!! YOU HAVE BEEN WARNED!!

Okay, I do sound a little cynical. I really shouldn't be complaining, as I don't have a ready solution for the problem.

Perhaps it is time to put on my Picard hat again, and make it so.

In the meantime, please remember to disable all unnecessary network services on every UNIX/Linux system under your control. If you are running a public Web server, set it up so it is ONLY a public Web server, and not a DNS server, POP server, IMAP server, FTP server, print server, rsh server (AARGH!), and so on. PCs are cheap, even if electricity is precious, so dedicate a system as a public Web server. And beware those self-destruct buttons.

In the meantime, please remember to disable all unnecessary network services on every UNIX/Linux system under your control

ISPadmin

DNS/IP Address Infrastructure

by Robert Haskins

Robert Haskins is currently employed by WorldNET Internet Services, an ISP based in Norwood, MA. After many years of saying he wouldn't work for a telephone company, he is now affiliated with one.

rhaskins@usenix.org

This installment of ISPadmin looks at ways ISPs design and implement their domain name system (DNS) infrastructure. For any service provider who has a range (or ranges) of IP addresses and/or domains allocated to it, DNS is at the core of the services offered. Just imagine the Internet today without DNS! IP address management and DNS are, by their very nature, intertwined.

Introduction

The domain name system's job is to map names to IP addresses and IP addresses to names. It works by delegating "zones" of data (namespace as well as IP space) out to the organizations who use it. The delegated nature of DNS makes management easy as the "owners" of the data are responsible for maintaining it. DNS is, by many accounts, the single most successful implementation of a distributed database.

The DNS protocol is defined by a number of RFCs; see the DNS Resources Directory for an excellent compilation of references (including RFCs) for DNS. The DNS-related RFCs (draft and standard) are far too numerous to list here.

For a small provider, a DNS design is likely to be relatively straightforward. The interesting DNS/IP address problem is for the larger provider, where more than two DNS servers are required. Also, a larger provider will likely have a much larger pool of IP addresses which require management.

The issue of DNS touches upon many areas, including:

- Billing
- NOC troubleshooting and maintenance
- IP address allocation
- Service delivery
- IP routing

While I will touch briefly on each of the above areas, I will focus on DNS deployment and architecture.

One might wonder what it takes to manage and support a typical DNS infrastructure. At Ziplink, about 500 domains were hosted and approximately 80,000 IP addresses (one per dial port) were managed by one staff member half-time. The server machines required for this infrastructure included three Sun Ultra 10 class machines including one shared master and two dedicated slaves/caches which handled both inbound and outbound requests. The shared machine (which had other services besides BIND running on it) handled all of the data for the DNS records Ziplink was authoritative for, feeding the two dedicated slaves/caches which responded to both internal and external DNS requests. The slave machines seldom ran at a load average greater than 1, and the load put on the shared machine by DNS was negligible.

Zone file record-keeping was a fully manual process at Ziplink, which accounted for the relatively large amount of time spent managing the DNS database. Many providers do not buy commercial tools or develop custom programs for managing their DNS records. If the provider does develop tools, they will likely not be very sophisticated and will require more manual data entry than a commercially available tool.

DNS Levels and Multiple Servers

There are several reasons why there are two classes or levels of DNS servers. The Internic requires two registered nameservers. Utilizing two DNS levels reduces the chance of errors as data is entered only once instead of twice. Also, this design allows for minimal impact to the "customer facing" (machines customers use for service) servers. Under BIND, each time a zone file is updated, the nameserver must be restarted. Utilizing a two-level design, the only time customer-facing servers are restarted is when a domain is added or deleted (i.e, a change to the named.conf is required).

In a perfect world, the two DNS servers would be on separate subnets fed by different routers in widely disparate geographical locations on the provider's network. Doing so would present the highest level of redundancy. This redundancy can be taken to very high levels. Imagine having multiple machines across your network with identical IP address(es), and by the magic of routing protocols be able to route to the closest one, even to another machine entirely if the closest one is down.

DNS for a Smaller Provider

Once again, the biggest issue driving a smaller provider is cost. As a result (and by virtue of the fact they are a small provider), at most, two DNS machines are usually deployed as depicted in (see Figure 1). In very small shops, they will be shared machines, which perform other functions (mail and/or RADIUS seems to be common).

One machine, labeled "primary DNS" in Figure 1, is where all changes are made to the zone files. Often, the provider will have written a script to assist in management of the zone data, and will utilize CVS or other source management tools as well. Some nameserver traffic will be pointed at this machine, but an effort will be made to ensure most of the load gets pointed at the machine marked "slave DNS." The word "primary" indicates the machine where zone data originates.

The machine marked "slave DNS" will usually be set up as a DNS slave or caching server, obtaining all of its authoritative data (zones about which the root nameservers query it) from the machine labeled "primary DNS." Doing so ensures the data is always in sync with the primary server, so there is no difference between what the two servers report.

In this setup, all DNS queries (both on and off the provider's network) are handled by both of the nameservers. Once the network is larger, this setup will likely change, and specific machines will be dedicated to inbound and outbound requests as outlined in the next section.

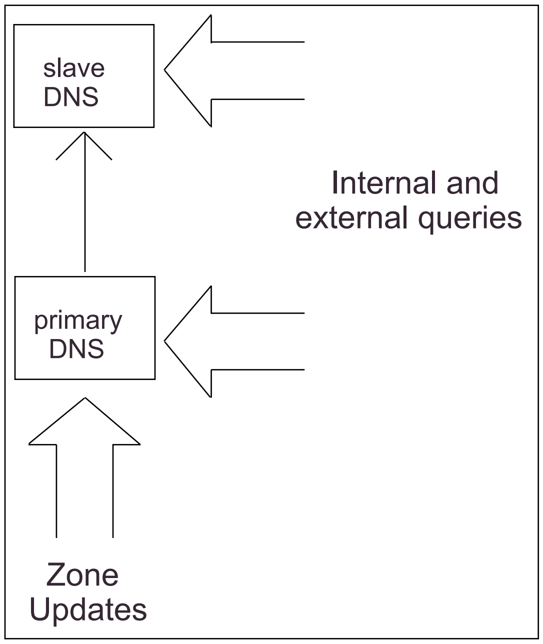

Figure 1

DNS for a Larger Provider

A larger network operator is going to be more concerned about redundancy and reliability than cost. As a result, they it will likely split their its DNS infrastructure into two pieces: one servicing internal requests (i.e., dialup ports, cable modems, DSL customers, etc.), and one servicing external requests (i.e., domains/IP addresses hosted by the provider). A bigger ISP might utilize the design shown in Figure 2 for their its external DNS traffic (requests originating outside the provider's network for domains/IP addresses hosted by the provider).

The machine marked "primary" in Figure 2 would be the single machine where all changes are made for which the provider is authoritative. No external requests would,

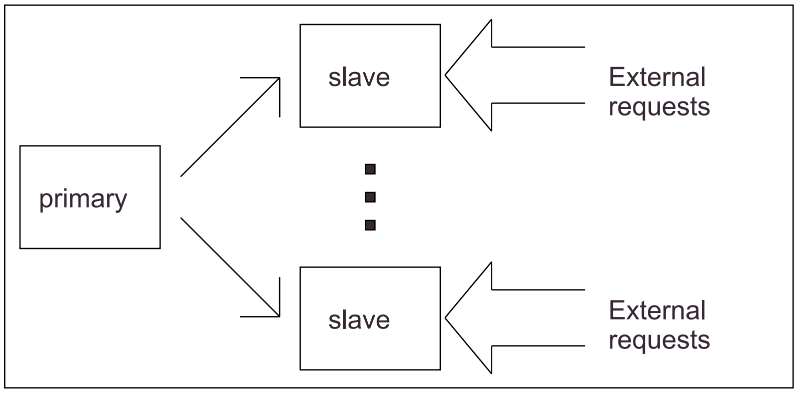

Figure 2

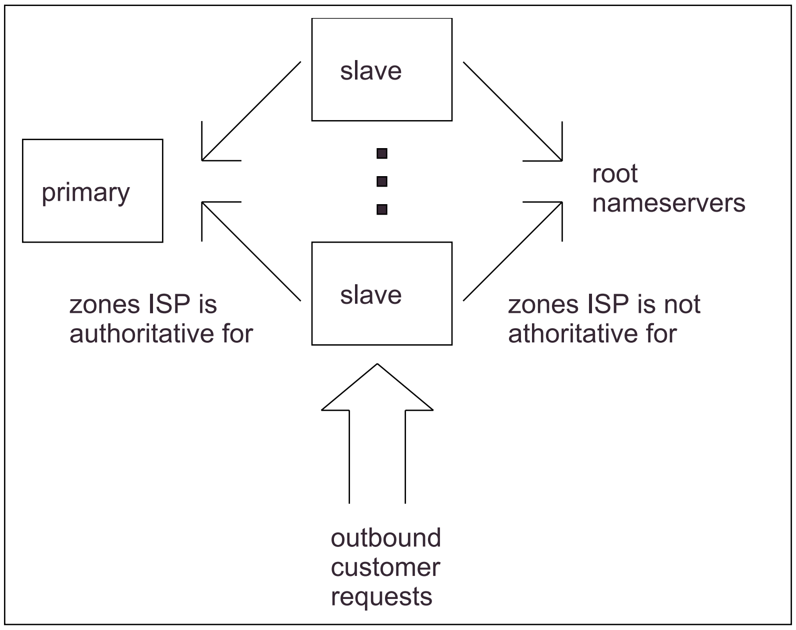
slave
primary
root nameservers
slave
zones ISP is authoritative for
zones ISP is not athoritative for
outbound customer requests

Figure 3

under normal circumstances, reach this machine. Its sole purpose is to feed data to the machines identified as "slave" which actually answer the queries coming in from networks outside of the provider's own networks. If you did a query on the root nameservers for data this provider is authoritative for, the machines labeled "slave" would show up. These "slave" machines' configuration would point to the internal machine marked "primary" in order to ensure they each reported consistent data. The "slave" machines would probably not have a pointer to the root nameservers, in order to encourage internal clients to utilize the caching/slave servers engineered expressly for this purpose.

Figure 3 illustrates how a larger provider might handle internal requests (name-service requests coming from its own "internal" network). Machines marked "slave" would be simple nameserver slave boxes, in the case of a dialup ISP deployed at the points of presence on the provider's network. The goal is to have the DNS servers as close to the end subscriber as possible. Of course, these caching servers would be like secondary servers in the sense they would be allowed to query the ISP's primary nameservers for zone data the ISP is authoritative for. Engineering DNS in this fashion enables fast access to all zones while reducing the load on the root nameservers to the extent possible.

DNS Server Software

The vast majority of ISPs, both large and small, utilize the Internet Software Consortium's (ISC) Berkeley Internet Name Domain (BIND) software. BIND has been around for many years and has been the subject of many security alerts. It would certainly be interesting to see some statistics on the usage of BIND and its alternative nameserver software, but I would guess the percentage of all sites on the Internet today utilizing BIND (or its derivatives) would be above 90%. If anyone has any pointers to such statistics, I'd love to hear from you.

BIND is considered the "reference implementation" for DNS, and the standard by which other nameservers are judged. While it has had its security issues (I am not aware of any security holes that have not been patched by the ISC), it does remain in wide use by the service provider community and in the Internet at large. The latest version of BIND is 9.1.2, which was released May 4, 2001. Quoting the ISC BIND Web site, "BIND version 9 is a major rewrite of nearly all aspects of the underlying BIND architecture." Check the ISC Web site for more information on BIND 9.

Most providers are running BIND 8, as BIND 9 will take some time to be "certified" and rolled into production. The process for certifying a new BIND version for production use could be something like the following (applicable to just about any new application in most information technology environments).

First, the provider will begin testing a new release of BIND in the lab for some period of time, enabling the staff to get familiar with the new features, bugs, etc. Once they are comfortable with the server and have come up with appropriate configurations for the production environment, a handful of low-use servers are upgraded for a few weeks. Finally, a complete rollout into all production machines is performed. All through the process, a way to get back to the previous version is preserved.

A couple of other DNS implementations bear mentioning. Perhaps the most well known is the djbdns server, by the author of qmail, Daniel J. Bernstein. Being aware of the security issues of BIND, the author has offered $500 "to the first person to publicly report a verifiable security hole in the latest version of djbdns." A less known server is Dents, an open source but not yet production-quality server. I am aware of a few providers who use djbdns, but none who are using Dents.

Another option for providers is to allow someone else to host their name service. A small provider might want to start by hosting their DNS records at a DNS provider while they focus on the rest of their business. Over the long term, however, most providers opt to host their own DNS as it is a critical part of providing Internet service. Perhaps for this reason, there are few commercial DNS service providers, and none whatsoever dedicated to the service provider market.

Namesecure is a commercial DNS service provider, but their initial focus was the name to dynamic IP resolution (for example, cable modem or server which connected via dialup for a few hours a day) resolution for end subscribers, not specifically hosting DNS services for service providers. Namesecure has since morphed into primarily a "value added" domain registrar similar to Verisign. Dynamic DNS is a free provider of DNS services, but again, their focus is almost entirely end users.

Interaction with ISP Operations

Most commercial ISP billing/provisioning systems and at least one free one (Freeside) I know of perform DNS provisioning by creating BIND-compatible configuration (`named.conf`) and zone files as part of their respective systems. This automation makes billing and provisioning DNS much more accurate and cost effective for the ISP.

The ISP's NOC personnel usually have access to the various nameservers to perform zone file updates and troubleshooting. This relieves engineering personnel from routine tasks and troubleshooting while giving the customer a better response time.

Network engineers at an ISP typically dictate how IP addresses are suballocated, once American Registry of Internet Numbers (ARIN) allocates a network to the ISP. Network engineering department input is usually required when provisioning new IP numbers or when setting up DNS name entries for network equipment.

Many ISPs in the recent past have shied away from allocating static IP addresses to customers due to the complexities of routing and managing this costly resource. Dialup ISPs associated with competitive local exchange carriers (CLECs) who are receiving reciprocol compensation from incumbant local exchange carriers (ILECs) encourage the use of static IP addresses. Static IP address customers tend to spend many hours online; the CLEC gets more money in the form of reciprocol compensation from the ILEC! I may cover the topic of IP addresses and related issues (ARIN, `rwhois`, IP address allocation/management, etc.) in a future column of ISPadmin.

Many ISPs in the recent past have shied away from allocating static IP addresses to customers due to the complexities of routing and managing this costly resource

Miscellaneous DNS Related Topics

DNS entries for the ISP's zone would vary depending upon the business plan and history of the ISP. Typical DNS entries for a dialup ISP owning the domain "isp.net" would be the following:

- www.isp.net – Web site for ~accounts
- smtp.isp.net – where customer outbound mail points to
- pop.isp.net – where customer POP clients point to
- pop3.isp.net – points to same IP as pop.isp.net
- mail.isp.net – points to same IP as pop.isp.net
- ftp.isp.net – anonymous FTP service, if provided by ISP
- news.isp.net – Usenet news machine(s)

Of course, using the magic of DNS round robin (or other load balancing mechanisms such as a layer 4 switch), multiple IP addresses can be returned for several machines providing duplicate services for redundancy or load purposes. A smaller provider would probably not have a need to do load balancing.

For hosted domains, the customer would dictate what entries should be placed into their DNS zone file. Of course, ISPs do not usually host DNS records unless the entity requesting the hosting has some sort of a business relationship with the ISP. Even with "secondarying" DNS records, usually the person requesting the secondary buys some sort of service from the ISP. There is at least one free public provider of secondary (and primary) DNS on the Internet called "The Public DNS Service" sponsored by register.com.

Network Address Translation (NAT) is a technique used by many organizations (especially enterprises) to reduce the number of IP addresses used. Typically, traditional ISPs are able to justify enough IP address space to cover their customer usage and do not deploy NAT as an enterprise would. An ISP's customer may need to deploy NAT because they doesn't want to pay the cost of additional IP address space, or the ISP doesn't have the space to allocate. Another way to reduce IP address usage is by utilizing Apache's (or other Web server's) virtual hosting capability. Name-based virtual hosting is the Web server's ability to serve multiple Web sites from one IP address. Utilizing name-based virtual hosting will drastically reduce the number of IP addresses required to serve large numbers of hosted Web sites.

Conclusion

DNS at the smaller scale is handled with two machines, a primary for making changes and responding to external requests, and a secondary for internalrequests. A larger network provider is likely to split up their DNS infrastructure: one machine to handle internal requests originating on its network and one to answer external requests not originating on its network, for from domains/IP addresses for which the ISP is authoritative. There are some free as well as commercial DNS service providers, but none aimed expressly at the service provider market. This requires most ISPs to implement and manage their own infrastructure.

Next time, I'll examine how ISPs large and small set up their Web hosting infrastructure. In the meantime, send your questions and comments regarding ISP infrastructure and system administration to me!

REFERENCES

DNS Resources Directory: *http://www.dns.net/dnsrd/*

ISC's BIND: *http://www.isc.org/products/bind*

Daniel J. Bernstein's djbdns page: *http://cr.yp.to/djbdns.html*

djbdns: *http://www.djbdns.org/*

qmail: *http://www.qmail.org/*

Dents: *http://sourceforge.net/projects/dents/*

Namesecure: *http://www.namesecure.com/*

Verisign: *http://www.verisign.com/*

Dynamic DNS: *http://www.dyndns.org/*

Freeside: *http://www.sisd.com/freeside*

ARIN: *http://www.arin.net*

The Public DNS Service: *http://soa.granitecanyon.com/*

NAT starting point: *http://linas.org/linux/load.html*

Apache virtual hosting page: *http://httpd.apache.org/docs/vhosts/*

a bit off the beaten track

by Strata Rose Chalup

Founder, VirtualNet Consulting; Strata Rose Chalup specializes in large-scale messaging deployment. A sysadmin since 1983, she is now enjoying a sabbatical to scuba dive, read sci-fi, fix her house network, and get enough sleep.

strata@virtual.net

I share odd bits of news with various folks, who often ask me, "Hey, where do you find this stuff?" I definitely use "the standard resources" that everybody uses, such as SlashDot, FreshMeat, and various OS-centric publications.

I'm also slightly tapped into the multimedia community, as well as the Web standards community, and it's here that I find a number of cool tools and interesting bits. Late, admittedly, by the standards of those communities, but perhaps early by the notice of the sysadmin and IT communities.

I find it worthwhile to wade through rather a lot of chaff to find the occasional sysadmin-relevant grain in the following places. They are NOT arranged by any order of importance.

Dave Farber's "Interesting People" (IP) [1 - 3 msgs/day]. Some cool tech, not restricted to computers, including inside track and commentaries on telecomm and privacy legislation, as well as forwarded news stories/URLs on those topics. Some very highpowered folks are on this list, and while only Dave can post to it, he frequently approves comments and replies mailed to him. Also covers computing history, usually in the first-person. Recent headers (as of this writing) include "UNIVAC turns 50," "Why we don't use digital cash," and "Feds will data tap under CALEA." Archives/page at http://www.interesting-people.org/

Geeks at Umich [Sporadic, 4 - 6 msgs/week usually]. An informal clan, centered loosely around the University of Michigan but scattered everywhere now. List topics include gadgets and goodies, forwarded IP bits of particular relevance to geeks, open source behind the scenes tidbits and/or choice forwards from the free software world. The latter is especially useful for me, since other than the occasional FreshMeat visit, I don't follow any dedicated news sites in that area. Topics occasionally include local events of interest in the Michigan area. Recent headers include "Unisys Apologizes for Creating Unintended Consequences of the Computer Age," "GODZILLA -> In a can!," "3-inch aluminum cube-o-fun," and "the eunuchs convention, june 20-29." That last was not about summer USENIX, it was a link to an actual eunuchs' convention in India. No public archives; subscribe to *geeks-request@monkey.org*, or via *majordomo@monkey.org*

Keith Dawson's "Tasty Bits from the Technology Front." [Updated frequently by blog (weblog), more sporadically by email]. A great source for breaking technology news, with a great deal of insider commentary, especially on ICANN atrocities, telecom policy, and major ISP outages. Other frequent topics include- cool gadgetry, science, and software tools, along with various odd soundbites. I am part of a small group of folks who have sent in tidbits to Keith, and we exchange lots of info on a private list associated with TBTF. All our really good stuff makes it into the blog, so you're not missing anything important! Recent topics included "An illegal prime number," "European Court of Justice outlaws criticism of EU," "When it absolutely, positively must be zapped overnight," and "A Bell goes south." Archives and blogpage at *http://www.tbtf.com/*

Phil Agre's "Red Rock Eater News" (RRE). [Sporadic, roughly weekly]. Primarily lists of one-line annotated URLs, labelled "pointers," on topics ranging from politics to science to technology to social systems. Monthly or so there are booklists of books, arranged by topic, that Phil has read, referenced, or just collected as relevant to the topic of the booklist. I am in awe of the degree to which Phil is well-read, and how many issues he tracks. Being a tenured professor has some advantages! Also includes occasional compelling forwards from educational, tech-educational, and public policy lists. The real gems, usually every 2 or 3 months, are first drafts and final versions of scholarly papers that Phil (and occasionally, list members) have authored on technology-and-society, economic theories of technology application, and the like. Fair warning: these can be very stiff reading, but are incredibly educational. Other content includes polemics and rants on the growing globalism, especially Davos and the like, American politics, university practices, lists of conferences, mostly on technology, crypto, education, CS, AI, and social sciences. Phil has a fetish for good, cheap pens, and you'll occasionally find indepth reviews of pens sent to him from odd places. Recent headers include "pointers," "Hague Conference: effects on free speech, consumers," "Hierarchy and History in Simon's 'Architecture of Complexity,'" and "The Information Society in Europe." Archives and Webpage at *http://dlis.gseis.ucla.edu/people/pagre/rre.html*

Need to Know (NTK). [weekly, on Fridays], In their own words, "*the* weekly high-tech sarcastic update for the UK." I love these guys! Lovely cryptic tidbits about the UK computer scene, including ISP foibles, legislation, and notably embarrassing gaffes and Web defacements, as well as updates on privacy and crypto legislation in the EU and overseas in general. Most issues feature a great tool spotlight, mostly free software but also commercial tools now and then. Other regular columns within issues include media updates, with info on UK TV movies and shows (with capalert links and highlights, and a popular new "junkfood" section about new varieties of nerdchow sighted and tried. Recent topics include "Guardian/Observer plugging 'dot-com-no-hoper' Moonfruit, why?", "Farewell to ORBS," "Review of ART DROIDS 2000AD museum exhibit, now showing," "Review of 'Get Over It' (Shakespeare in Love meets Bring It On)." Archives and Webpage at *http://www.ntk.net/.*

Dave Winer's "Scripting News Update." [daily, sometimes highly prolific]. More of interest to the Web world types, but I work there and try to keep up on what's going on. Mostly XML, RSS, Web standards, and Web industry news, announcements of related packages and tools, interesting bits of overall computing industry news, and opinion pieces by Dave. The latter are usually either completely cryptic or utterly fascinating to me, given that I don't overlap with that world very much. One interesting thing that shows up frequently is first-person quotes from industry luminaries on a variety of topics, since Dave has been around forever and knows everybody. Recent topics included "Microsoft-Free Fridays," an announcement of new Perl XML-RPC implementation, "Google Buttons," and many pointers to informative rants and amusing satires on Smart Tags (by multimedia industry software developers). Archives and Webpage at *http://scriptingnews.userland.com/* It's worth noting that UserLand is a freely available scripting publishing environment that Dave is the primary author of.– if Wiki or Blogger don't quite do it for you, try UserLand.

David Weinberg's "Journal of the Hyperlinked Organization." (JOHO). [monthly] Primarily Web development and standards community news and tool announcements, quite highly thought of in that community, with quality and in-depth writing. Recent topics included "save the threads," "Breaking the spine of books," "The three-strikes rule for PR," "Data spidering service," and "Building a fullsize Robbie the Robot replica". Archives and Webpage at *http://www.hyperorg.com/*

Owen Thomas' "DITHERATI." [Daily, weekdays]. In their own words, "see the digerati dither, daily." Pompous, ludicrous, and ridiculous quotes by "industry luminaries" who ought to know better, captured and usually poked at with a verbal stick by the good Mr Thomas. Includes the source of the sighting and a brief, usually stinging, comment on the disparity between the quote and reality. Since I don't follow the "dot com world" or "the industry" too closely, I find this to be a useful remedial education as to the major players and companies, as well as highly amusing. Recent topics included "A FISH, A BARREL, AND A SMOKING GUN," "Internet: not free Phone monopoly: free, Any questions?," "Neither rain, nor snow, nor gloom of five-hour compiling sessions," and "JEWEL IN DENIAL." Archives and Webpage at *http://www.ditherati.com/.*

Glen McCready's "0xDEADBEEF." [sporadic]. Moderated, primarily humor and the occasional highly nifty scientific bits. Some of the humor is not "family friendly," though it's usually technically "clean." Subscribe at *0xdeadbeef-request @petting-zoo.net*

The Conversation Continues. [monthly] This is a free newsletter put out by Esther Dyson and staff, and is one of the few "pro" sources I receive. It's a combination of informative teasers about articles in Release 1.0, Dyson's highly respected but expen$ive subscription newsletter, and fascinating op-ed and industry news pieces from a very interesting perspective, namely mover-and-shaker VC community folks engaging in prolepsis trying to predict the future of "the industry," or at least new trends and hotspots. Much of it seems pointless or bizarre from here in the quasi-trenches, but bear in mind that your CTO and his/her buddies are probably reading Dyson's for-pay newsletter, and that Gartner is influenced by Dyson, generating similar stuff in their "vision" subscription sections. Fear. In some ways it's like watching a road accident, there's that same fascination and horror upon watching tomorrow's IT crises being created out of air and dew by very savvy and usually accurate business computing policy experts. Also serves up some ICANN and related news/views since Esther has been heavily involved there. Recent topics included "ICANN Wants *You*!!!," "Triumph of the Weblogs (teaser)," "Feedback and Further Conversation," and a calendar of Upcoming Technology Events. Archives and Webpage at *http://www.edventure.com/conversation/.*

There are a lot more, but these are just the ones I make time to read every few weeks, or more often. Then there are the ones I save for a two or three months, or when I have spare time (ha!) or when I'm procrastinating.

Send me your favorites, and I'll create a page for the list and announce it to the sage-members, and as an addendum to a future column.

the corporate policy web

Summary

Every company has policies and procedures designed to reduce mistakes, increase the likelihood of consistent behavior and generally minimize risk. The goal of this column is to list some of the policies that you and your general counsel should work together to develop and maintain. In general, your company should probably have the following policies: an Acceptable Use Policy for internal users, a Terms of Service for any external users/customers, a Monitoring Policy, a Data Retention Policy, an IT Risk Management Policy, an Incident Response Policy, and a Privacy Policy. Each policy should be tested/audited on a regular basis.[1]

by John Nicholson

John Nicholson is an attorney in the Technology Group of the firm of Shaw Pittman in Washington, D.C. He focuses on technology outsourcing, application development and system implementation,

John.Nicholson@ShawPittman.com

Introduction

Corporate policies and procedures serve a number of functions – they are educational, they increase the likelihood that processes will be performed consistently, they minimize the number of mistakes made or steps skipped, and they provide the company with a document that it can show the world to say "This is how we do 'x'." Developing corporate policies also allows your company to figure out potential responses to situations ahead of time, decreasing the number of decisions that are made in the heat of the moment.

In several other columns, I've discussed some of the policies that your company ought to have in place: a Harassment Policy, an Acceptable Use Policy, a Monitoring Policy, and a Security/Risk Evaluation Policy. The purpose of this column is to add to that list and explain why this collection of policies is necessary and what they should look like.

Policies Your Company Should Have

As far as your company's use of technology is concerned, your company should probably have, in some form or another, at least the following policies in place and under regular testing and compliance review:

- Acceptable Use Policy for internal users
- Terms of Service for any external customers/users
- Monitoring Policy
- Data Retention Policy
- Risk Management Policy
- Incident Response Policy
- Privacy Policy

Acceptable Use Policy

The Acceptable Use Policy (AUP) is the terms of service by which all employees, contractors, etc. (including executives and administrators) access and use your network and internal systems. Your AUP should govern all of the systems that an internal user might use, including corporate systems, email, intranets, corporate databases, peer-to-peer applications such as Napster and Gnutella, PDAs, personal ISP accounts, instant messaging software, and anything else they might be able to use or access via your network.

An AUP educates users about what they can and cannot do, and informs them of any penalties that may be associated with doing things they are not allowed to do (e.g., account termination, disciplinary action, termination of employment). An AUP should include at least the following:

From a legal perspective, an AUP can help limit the company's exposure to harassment or breach of confidentiality claims

- It should specify that your network and systems are for business purposes and that personal use of the system is not permitted or is strictly limited (and if limited, how).
- It should specify that all data on equipment provided by the company is the property of the company and may be inspected at any time.[2]
- It should reference your Monitoring Policy and state that, as a condition of access to the network and systems, the users consent to such monitoring.
- It should include a provision that specifies that even if your company does not take disciplinary action regarding a particular unauthorized use of the network or a system, such failure by the company to take any action should not be interpreted as a change to the AUP or permitting such unauthorized use in the future.
- It should prohibit the use of the network and systems for any illegal act or breach of regulations (including those related to intellectual property, anti-hacking, anti-fraud and/or data privacy) or company policies, including, in particular, your Harassment Policy and your Privacy Policy.

If you provide services to external users and have a Terms of Service (TOS), your AUP should specifically require your employees and contractors to comply with the TOS when using those services in the way that a customer would.

From a legal perspective, an AUP can help limit the company's exposure to harassment or breach of confidentiality claims. The AUP also establishes the boundary for "lack of authorization" for purposes of the Computer Fraud and Abuse Act.[3]

The AUP should be regularly updated to cover new technologies. Employees should have the AUP explained to them as part of their orientation. The AUP should be included as part of your employee handbook, and a condition specifying that contractors will abide by the AUP should be included as part of any contracts with consultants or other third parties who might have access to your network or systems. Also, all employees (and contractors who will have access to your network or systems) should be required to sign a statement that they have received, read, understood, and agree to comply with the AUP. It might also be a good idea to do something to regularly remind people about the AUP such as posting it on physical and electronic bulletin boards or including a message regarding compliance as part of login banners or as a click-through screen as part of the login process.

Terms of Service

The Terms of Service (TOS) governs the use of customers or external users of your network or systems. Your TOS should:

- Clearly describe the service provided
- Claim ownership of the intellectual property included as part of the service
- Grant users a license to the intellectual property necessary for them to use the service
- Grant a license from the user to you for your use of any intellectual property provided the user as part of the user's using of the system
- Prohibit users from violating laws or regulations or performing any other acts that your company wishes to prohibit
- Specify whether or not users are allowed to link to your service and, if so, any restrictions on such linking
- Explain your Monitoring Policy and specify that, as a condition of accessing your network, systems, or services, the user consents to such monitoring

- Indicate that the user accepts the TOS and will use the service in accordance with it

If users access your service via a Web page or some other type of login screen, the TOS should be a click-through screen, preferably with the button at the bottom of the page. That way, users at least have to scroll through the whole TOS before clicking the button to accept.

Your TOS may be subject to certain legal requirements. For example, under the Digital Millennium Copyright Act, a Web site's TOS should include a contact for copyright violation claims. If your Web site, or any part of your Web site, is intended for children under the age of 13, there are very specific rules governing how you provide those services and what information you can collect. You should coordinate the development of your TOS with your company's general counsel.

Your Monitoring Policy should specify that your company has the right to monitor all network traffic and all data stored on equipment used for company purposes

Monitoring Policy

Sections 2511 and 2520 of Title 18 of the US Code create criminal and civil liability for improper interception of wire, oral, and electronic communications. Although there are exceptions under both the US Code and under state laws for system providers, relying on these exceptions is unnecessary if your company puts in place an appropriate Monitoring Policy. By explicitly requiring user consent to monitoring, your company can make access to your network and systems conditional on users accepting such monitoring. All users of your network and systems (whether employees, third-party contractors or customers) should be required to consent to monitoring.

Your Monitoring Policy should specify that your company has the right to monitor all network traffic and all data stored on equipment used for company purposes that is provided to an employee or contractor by the company or by any third-party contractor. Both your AUP and your TOS should reference this policy and explain it. Login banners should also reference the Monitoring Policy and state that access to the network or system is subject to monitoring at any time and for any reason, and that by accessing and using the network or system, the user is explicitly agreeing to such monitoring.

Monitoring traffic and behavior on your systems can allow you to detect misconduct in real time and can create logs that will be useful in an investigation and/or prosecution. Monitoring can also decrease employee Web surfing or other violations of the AUP.

In the future, the increased use of personal technology (e.g., cell phones, PDAs, etc.) to access corporate systems will require increased and more specific consents. If, for example, you open up your document management system so that it is Web accessible, an employee with a PDA and a wireless modem can download confidential information. Access to that system could require explicit consent from the user to monitoring of the activity and an agreement to provide access to the PDA on demand. (Note, such access will be easier if your company owns the PDA and provides it to the employee.)

Data Retention Policy

Your Data Retention Policy (DRP) may already exist. Frequently, companies have policies that specify how long paper records will be retained, both on-site and off-site, and what will be done with them after that period. They might be microfilmed, sent to a warehouse or just destroyed. Depending on your industry, it's even possible that your

Given the possible regulatory aspect of data retention, as well as the possible use of stored data in litigation, your general counsel should be involved in the development of your DRP

DRP is mandated by some regulatory agency. It's also very likely, however, that any existing DRP is already being violated by your computer users. If your DRP was developed prior to the widespread use of PCs, your DRP probably isn't even suited to dealing with electronic data.

Given the possible regulatory aspect of data retention, as well as the possible use of stored data in litigation, your general counsel should be involved in the development of your DRP.

Your DRP needs to deal with both paper and electronic data and must comply with any regulatory requirements imposed on your industry. Given the ease with which documents are now generated, you may even need to figure out how to deal with multiple copies or versions of both paper and electronic documents. Whatever policy your company decides to impose, the DRP and its implementation procedures should be clearly communicated to your users. The implementation of your DRP should also be audited.

IT Risk Management[4]

Your IT Risk Management Policy should be part of your overall corporate Risk Management Policy (assuming you have one). Your IT Risk Management Policy should include a procedure that tracks security risks (both external and internal) as they are identified, evaluates their potential risk to your business, identifies the appropriate fix, schedules a date for the implementation of the fix, and includes a follow-up procedure to ensure that the fix was properly implemented. For example, your policy should include:

1. Regular reviews of the relevant security vulnerability sources (i.e., Bugtraq, NTBugtraq, security reports published by software vendors, virus reports, security researchers, the various cracker Web sites, etc.) and, if appropriate, a procedure to ensure that such reviews are performed

In a diverse environment, your company may have multiple people responsible for various platforms and/or software packages, or your company may have various administrators with responsibility divided by geography. It's important to make it clear who will have the ultimate responsibility for monitoring security issues related to each platform or software package.

2. A determination of how the identified vulnerability applies to some aspect of your business

For example, a security hole that lets a script kiddie put graffiti all over your Web page can be embarrassing to your company or might result in your taking down the page until you can plug the hole. If your Web page is just information about your company, this might not be a big problem. If your Web page is the means by which your customers order, that's a different matter. It's important to understand how the vulnerability could impact your business if it were exploited.

3. A rating of the risk represented by the security issue (i.e., Critical, High, Medium, or Low) based on the potential impact of the security issue to the business (in terms of lost business, lost data (based on your DRP), public perception, potential cost, etc.)
4. A schedule for the implementation of the relevant fix for the risk (i.e., all Critical fixes will be implemented within one day, all Highs within one week, etc.)

5. A follow-up procedure that checks whether fixes were actually installed and, depending on the importance of the security issue, verifies whether the fix actually solves the problem

A follow-up procedure could vary depending on the rating of the issue. For example, you might want to ensure that all fixes for critical issues are implemented, and use statistical sampling for the remaining fixes. Alternatively, you might want to ensure that, regardless of rating, all fixes for a mission critical system are performed.

Finally, your policy should schedule regular audits of how your system stacks up against the known threats. This might involve having a "white hat" security firm attempt to penetrate your network. Such audits are an opportunity to test your priority ratings, as well. If a problem someone rated as "Low" allows the penetration team to take control of your system, then you might need to reevaluate that rating.

Your response to an incident should never be ad hoc

Incident Response Policy

Your response to an incident should never be ad hoc. Depending on the nature of your business and the type of incident, the personnel involved should have a clear plan for how to respond and whom to (and not to) inform (both internally and externally). Your Incident Response Policy (IRP) should be developed by a multidisciplinary team that includes (1) knowledgeable representatives from IT, Security, Legal, PR/Marketing, and Insurance/Risk Management and (2) selected third parties, including forensic experts, security consultants, and possibly law enforcement.

The IRP should identify initial indicators ("triggers") of an incident (obviously these must be updated regularly) so that those involved know when to initiate the response plan. Different indicators may require notice to specific people or certain actions. Regardless, such notice and actions should be precisely scripted. As part of the development of the IRP, your team should think through various scenarios and plan first responses to each of them. You should also identify in advance those external parties (preferably specific individuals) who will provide support during different types of incidents. For example, you might identify specific technical and forensic consultants, certain local or federal law enforcement officers, individuals at your ISP, external legal counsel, a crisis management firm, etc. These people should be included in the development of relevant sections of your IRP.

In developing your IRP, your team should have regular meetings with IT staff. Members of your incident response team should not be meeting each other for the first time when you have an incident. It is important for your team to understand who has access to your systems, what is the extent of their authorization (in particular and as specified in your AUP), what type of logs or backup copies are available (as specified in your DRP), what range of response and notice options are permitted by internal policies, and any external requirements.

ELEMENTS OF THE IRP

Trigger Events

This section of the IRP should identify the triggers for initiating the IRP. Such triggers might be based on particular networks or systems, data or events. Some triggers might result in the IRP being implemented automatically, others might require evaluation by designated parties. As part of identifying the triggers, your team should have a good understanding of the steady state "background noise" of the network and systems. The triggers should not be set so low that the IRP is always "crying wolf," but, at the same

Your IRP should clearly specify how evidence related to an incident is to be maintained and protected

time, they need to be sensitive enough to initiate the IRP when appropriate. One approach when implementing a new IRP would be to set the triggers very low and gradually raise them as the team and your company gain an understanding of which events actually constitute threats.

Incident Evaluation

Once an incident has been identified, your IRP should specify how that incident should be evaluated. The IRP should require those working the incident to prepare answers to questions that will be relevant to decision-makers. For example:

- Does the incident appear mischievous or malicious?
- Does it appear to be an isolated incident or part of a larger pattern?
- If the incident is network-based, is the upstream source more likely a victim or the originator of the incident? If not network-based, what is the source?
- What are the implications of contacting the source?
- Has your system been compromised? If so, where and for how long?
- What systems/files have been taken/tampered with?
- What is the value/potential harm resulting from such tampering/taking?
- From a legal perspective, does the incident create potential liability to customers, shareholders, or other downstream entities? What level of due diligence is required to avoid liability if the incident escalates?
- If there is an investigation, should participation in the investigation be limited to internal personnel? Should outside counsel be retained?
- Does the company have reporting obligations related to incidents of this type? If so, to whom?

Evidentiary/Forensic Requirements

Your IRP should clearly specify how evidence related to an incident is to be maintained and protected. You should work with your general counsel, law enforcement, and external forensic consultants in advance to develop this portion of your IRP. Proper evidentiary and forensic procedures will increase the likelihood that your company will be able to recover any damages and that an attacker will be prosecuted. These procedures do not kick in until after an incident has been resolved. They are an integral part of the incident response process.

Responses

Your possible responses to an incident range from ignoring it to immediately shutting down the affected system. Your IRP should specify under what circumstances you will ignore an incident, when and for how long you will allow it to continue while you observe and gather evidence, when you will take immediate action and, if so, what action you will take. Without the guidance of an IRP, the immediate response will probably be "Shut it down." Shutting it down lets an attacker know that you are aware of the problem and allows the attacker to move on and attack you in another way or attack someone else. Shutting it down might also compromise evidence, thereby preventing you from prosecuting an attacker.

In the long run, your responses to an incident should be tied into your IT Risk Management Policy. The positive side of an incident (if there is one) is that an incident will help you evaluate whether your rating of a particular vulnerability was correct.

Reporting Procedures

Your IRP should also specify what internal and external reporting will be done regarding an incident, including if and/or how an incident should be reported to law enforcement. The IRP should designate who has the authority to make any external reports regarding an incident and to whom they are authorized to make such report. For example, your risk manager or your general counsel may be authorized to call in law enforcement, but only the CEO would be authorized to communicate with shareholders and the media.

An important area to consider in the development of your IRP is under what circumstances you wish to involve law enforcement and at what level. Should it be the local law enforcement, or should it be the FBI? Either way, you should have an established relationship with whomever it will be. Bear in mind that once law enforcement is involved, the handling of the incident may be out of your control. Law enforcement may have a different agenda than you do.

Testing

Once you have your IRP in place, you should test it on a regular (and occasionally unexpected) schedule. You should also consider bringing in all of your external support for an exercise from time to time, so that you can see how your IT, legal, forensic, PR, and other experts interact. The likelihood of a significant incident and the potential impact to your business of that incident should determine how frequently and to what degree you conduct such tests. Such testing can be tied into testing of your IT Risk Management Policy.

Privacy Policy

These days, everyone is concerned about privacy. If you are in certain industries (finance or health care) or if you provide services online to children under 13, you are subject to very specific regulations regarding any information you collect online. If your business operates in Europe, you may be subject to the European Data Privacy regulations. Canada recently passed a data privacy act. And more privacy legislation, both in the US and abroad, is coming.

If you collect and store any personal information from customers, members or anyone else, you should have a Privacy Policy that clearly describes what data you collect, what you do with it, how people can opt out of having you keep or use their data and how they can contact you to correct any errors or opt out. If you have a Web site, there should be a link on each page of your Web site to the Privacy Policy. The Privacy Policy should clearly include a mailing address, email address, or phone number where people can contact your company regarding privacy questions.

Your Privacy Policy should be tied into your DRP. You should also consider in your IRP whether and/or how you will notify people if their data is compromised.

Once you have your Privacy Policy in place, your compliance should be audited on a regular basis by both your internal audit staff (if you have one) and your external auditors.

If you collect and store any personal information from customers, members or anyone else, you should have a Privacy Policy

Conclusion

Developing and maintaining this collection of policies is a lot of work. It requires coordination between technical and legal personnel, and the policies require testing. It's not enough to simply write a policy and then put it in a drawer. Once your draft policies have been developed, have them audited on a regular basis by your internal audit group (if you have one) and/or your external auditors. In order for your policies to do their job, everyone whose behavior is governed by a particular policy needs to understand and comply with it. Properly implemented, a good set of policies can substantially decrease both the operational and legal risks to your company. Doing it right is expensive and resource intensive, but the risks associated with doing it wrong or not doing it at all are too high.

NOTES

1. This article provides general information and represents the author's views. It does not constitute legal advice and should not be used or taken as legal advice relating to any specific situation.

2. Given the increasing capabilities of PDAs, there is an increasing likelihood that a company will, at some time, want to see the contents of an employee's PDA. If the PDA belongs to the employee, the company may not have the right to demand the contents of the PDA. If, on the other hand, the company has provided the PDA to the employee, then the company has the right to look at the data on the PDA at any time, just as if it were on a company-provided computer.

3. 18 USC § 1030.

4. Note, this section repeats information provided in "You've Been Cracked. . . And Now You're Sued," in the April 2001 issue of *;login:*.

turf

This article continues the previous theme of highlighting the similarities between organizations and other living organisms. Because we are primates, it is quite reasonable to expect that organizations made up of people will behave, in some ways, like primates.

by Steve Johnson

Steve Johnson has been a technical manager on and off for nearly two decades. At AT&T, he's best known for writing Yacc, Lint, and the Portable Compiler.

yaccman@earthlink.net

and Dusty White

Dusty White works as a management consultant in Silicon Valley, where she acts as a trainer, coach, and troubleshooter for technical companies.

dustywhite@earthlink.net

One of the most obvious similarities relates to marking and defending territory. Primates, and many other creatures, find some "turf" which they claim for their own. They then mark this territory and defend it against encroachment. Organizations do too.

This is often evident in the staff side of a company. The travel department goes ballistic if you order your airline ticket on the Internet. The finance department wants to see and approve every check. The facilities department is upset about nonstandard furniture. MIS gets snooty about that "legacy" peripheral you want to connect.

We aren't saying this is right or wrong. These responses may be quite rational, or totally irrational. The point is that we are probably wired to have them, at least as individuals, and this means that organizations tend to have these responses as well.

Sometimes the territory wars are very obvious — who has the biggest office, the corner office, the office across from the restrooms, the office with a pole in the middle and a busted thermostat? Who is in a cube? Is it a tall-walled cube or a low-walled cube? Does it have a new chair or a cast-off one? And so on...

We have a tendency to respond to territory emotionally, as anyone who has ever planned a building move knows in spades. So how can we allocate space and keep ourselves out of the "red zone" of irrational behavior. Sometimes space is allocated hierarchically — a company gets a building, each department has a floor, each group has a separate corner, and then the groups allocate their own offices. This isn't a bad way to allocate space, although sometimes the fights over which floor and which corner can blindside you by their virulence and irrationality.

We suggest a way to approach space planning that has worked well for us in many diverse situations. In fact, we will give a couple of simple consistency proofs supporting this method.

The method is the following:

1. Rank the employees.
2. Pick the offices by rank. The highest-ranked employee gets the first pick, the second-ranked employee the next pick, and so on.
3. Bend the rules a bit to account for those with special needs, doubled offices, major telecommuters, and so on.

The two obvious ways of ranking are: "by seniority or experience" and "by random draw."

Before discussing this further, let's assume that there are N people being put into N offices. Further, let's assume that everyone agrees on the desirability of the offices – 1 is the best, down to N, which is the worst.

Now, the question we pose is: "What story do you tell the person who is put into office N?" If you have used seniority or experience, the story you tell this person is "You have the least seniority or experience, so you get this office." If you have used a random draw, you tell that person "You were very unlucky."

Almost every other scheme I am aware of sends a worse, and much more personal, message to the employee in office *N*, namely: "A bunch of powerful people got together and decided that you are the least deserving person here, so you get the worst office." Is that unmotivating, or what? There is no particular stigma over being inexperienced or unlucky, but the notion that your superiors consciously assigned you to a black hole is not going to make your day.

Once the employees are ranked, it is possible to have a "party" with everyone present to pick the rooms. It's surprisingly pleasant, with much groaning and cheering. The later pickers have a context and know who their neighbors will be, which helps offset the less desirable choices.

We personally favor seniority, as being more deterministic than a random draw. In fact, there is a simple argument that supports seniority. Suppose we have *N* offices as before, but only *N* - 1 people. Then when the dust settles, office *N*, the worst one, will be unoccupied. Now, suppose a new person is hired. The obvious place to put them to minimize disruption is in office *N*. And this is just where the seniority scheme would put them. So this scheme is stable when you hire people, which is a good thing.

Whichever scheme you use, remember that you are experiencing the organization as primate, and get ready for some deep and irrational feelings to surface in yourself and others.

the bookworm

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He is Chief Knowledge Officer at Matrix.Net. He owns neither a dog nor a cat.

peter@matrix.net

The USENIX Conference in Boston finally convinced me that folks read this column. Oh, I'm not trying to be coy: whenever there's a glaring error, I hear it. But as I walked the exhibit floor in Boston, people I hardly knew (or didn't know) would ask me about this book and that. It's really flattering. So is the response I've had to my call for more reviewers. I now have a group of volunteers who will be doing reviews. Just how many and how frequently will be a function of the topics they have an interest in and what gets sent in by the myriad publishers.

I am especially pleased that I've found volunteers in Canada, Germany, and Italy, spreading our scope geographically.

And now for the autumn's books.

Berkeley DB

Databases are important. Embedded systems are important. The Berkeley database is the most widely used embedded database system in the world. The more we use embedded databases (as every time you employ Netscape or order a book from Amazon.com or use a handheld device), the more important understanding them becomes.

Berkeley DB is divided into two parts: the first, pp. 1–242, is a reference manual of great value; the second, pp. 243–632, is the API manual. The latter details the APIs for C, C++, Java, and Tcl. The book concludes with a section on supporting utilities and an excellent index. (NB: If you just want to use a database, this book is not for you. If you are a programmer with at least some knowledge of databases, this book is for you.)

Berkeley DB is a good book on a first-rate, open source database. The only criticisms I have are of the volume's production: first of all, the page numbers in the table of contents bear only a tangential relationship to the actual chapters (luckily, the index was done by reliable software); secondly, two figures have their labels reversed.

The folks at Sleepycat Software have done a great job: Margo, Keith, Mike, Mike, and whoever else was involved in this, my compliments.

Being Disruptive

Over the past 5000 years, most media have functioned on a one-to-many basis. The massive temple inscriptions, the imposing stelae of the Babylonian, Egyptian, and Persian empires bear testimony to the beginnings of this: "I, Darius, great king, king of kings..." begins column 1 in Behistun (parodied by Shelley in "Ozymandias" [1818]). The sacred books of all religions are proclamations from the few to the many. So, in more recent centuries, the book, magazine or newspaper publisher, the radio and the TV broadcaster all operate on a one-to-many basis.

From its very beginnings, the Internet has broken this model: every machine on the Net peers with every other. Even when there were but a dozen or a few hundred hosts, there was no notion of publisher/source and passive receiver. As we're now at over 150 million machines on the Net, "closing down" the publisher or broadcaster (a popular pastime of oppressive regimes) has become truly impossible.

BOOKS REVIEWED IN THIS COLUMN

BERKELEY DB
Sleepycat Software, Indianapolis, IN: New Riders, 2001. Pp. 664.
ISBN 0-7357-1064-3.

PEER-TO-PEER: HARNESSING THE POWER OF DISRUPTIVE TECHNOLOGIES
ANDY ORAM, ED.
Sebastopol, CA: O'Reilly, 2001. Pp. 432. ISBN 0-596-00110-X.

BEYOND CHAOS
LARRY L. CONSTANTINE, ED.
Boston, MA: Addison-Wesley, 2001.
Pp. 416. ISBN 0-201-71960-6.

CERT GUIDE TO SYSTEM AND NETWORK SECURITY PRACTICES
JULIA H. ALLEN
Boston, MA: Addison-Wesley, 2001.
Pp. 447. ISBN 0-201-73273-X.

A SYSTEM ADMINISTRATOR'S GUIDE TO AUDITING
GEOFF HALPRIN
Short Topics In System Administration, 6.
Berkeley, CA: USENIX Association for SAGE, 2000. Pp. 52. ISBN 1-880446-21-9.

By and large viruses or worms or DDoS attacks are just annoyances, pranks. But for over 150 countries, the Internet has become a road to news that does not pass through government control, a method for nearly anyone to both send and receive at will.

Andy Oram has put together an anthology of pieces on technological, legal, financial, and social repercussions of peer-to-peer Internet communication. This goes far beyond SETI on the one hand and Gnutella on the other. The mere existence of anonymous remailers (even after Julf Helsingius shut his down) frightens the thought police.

Publius tells us about trust. Red Rover tells us about really low-tech distribution. The book contains 19 essays and an afterword.

Like any anthology, the quality is uneven. But it's worth reading (and thinking) about.

Software Development

For four years, Larry Constantine ran/edited/wrote a "forum" in *Software Development* magazine. Forty-five of the columns (by a large variety of folks) have been collected in *Beyond Chaos*. Most of the essays are interesting and, thanks (I suspect) to editing by Larry and the magazine's staff, quite readable. I found several very illuminating; a few (in retrospect) seem just worthless; but the just over 400 pages are great for reading on a flight, at the beach, or wherever. The essays are brief and thus the volume can be read in snippets.

I found it heartening to realize that Aristotle is still relevant today. Larry states (Chapter 31): "The artist learns how to paint by painting." Aristotle wrote: "He who learns to play the harp learns to play it by playing it" (*Metaphysics* 1049b31f). There's something similar in the *Nicomachean Ethics* 1103a32–34.

John Boddie's Chapter 10 is a keeper.

CERT's Practices

With input from a large number of folks at the SEI and at CERT, Julia Allen has produced a simple, practical guide to protecting your system(s) from unauthorized intrusions. My guess is that many readers of this column will find the book too simple, but it seems to me that with the Internet and systems growing at a furious rate, the number of experienced sysadmins is waxing far too slowly. There are thus a number of folks who need a milder, more basic approach. There are also a number of people who work in environments where the highest levels of management don't understand the details. Here is a book that carries CERT's authority to hammer them with.

A Major Omission

I owe Geoff Halprin an apology. It's a year since his SAGE booklet appeared, and I've neglected it. I could offer excuses, but instead, I'll just give him a few flattering lines.

There are all sorts of jokes about how dull auditors are. Computer auditing has never, I admit, appeared a fascinating topic to me. But Halprin's 50 pages convinced me that the "rigorous examination of a system" together with the "identification of shortfalls in compliance or practices" and the "organized repair" of the system are indeed very important. Good job, Geoff. And, again, my apologies for taking so long to print these few words.

USENIX news

Electronic Property

by Daniel Geer

President, USENIX Board of Directors

geer@usenix.org

The old advice about never discussing religion or politics at family gatherings notwithstanding, it is presumably unarguable that the percentage of the world's valuable things that are electronic in form is increasing. The "market share" of electronic goods is rising. Taking these electronic goods as a form of wealth, we here are some mix of creators, custodians, shepherds and guards. Those of us specifically in security are also professional paranoids.

USENIX repeat customers, i.e., USENIX members, tend to be in favor of sharing information. Indeed, the old mission slogan for USENIX, "Moving information from where it is to where it isn't" remains in full force even if it is more commonly practiced than recited (the sign of a powerful idea, as ever). To use a food analogy, USENIX lays on one of the best tables of information in the business. From that, we derive our ability to do it again. And again.

USENIX as an organization has a voice, and just like an opera singer, our voice needs to be used just enough to keep it powerful but not too much. We are asked, more often than you might think, to support this or that cause with our (your) voice and with our (your) money. I've spoken gently in these pages before about the use of (your) money and look forward to the day I can debate the issue of non-program expenditures of program-derived funds at full vigor and without regard to the diplomacy my office requires. Today, however, I want to speak about the use of our voice.

As you doubtless know, USENIX is by the (US) tax code a so-called "501(c)3" organization incorporated in the State of Delaware. Such an organization is not only non-profit and tax exempt in and of itself, but gifts to it are tax deductible to their donor. Such a designation is non-trivial to get but trivial to lose – just stop being non-profit, stop playing by the myriad rules for non-profits, or use tax exempt monies for purposes that are officially forbidden. Such forbidden uses include lobbying for legislation, endorsing candidates for public office, advocating positions that are unrelated to the formal mission of the organization be it a charitable organization, an educational organization, or any of the other specific sub-species of the "501(c)3" genus.

In the matter of information sharing, we (USENIX) try very hard to make sure that the information that we share is with the permission of those who own it. Note that I say "own" as information is as much a valuable good and subject to ownership as a pair of shoes, a movie ticket, or a pint of Guinness. There is a widespread, anthropomorphic, pseudo-moralistic argument that "information wants to be free." Perhaps, but in that sense fire wants to burn, rain wants to fall, and entropy wants to work itself out of a job.

No, "information wants to be free" is a falsity but it is, and with abundant evidence, more than true that in the electronic sphere the idea of property takes on a whole new set of axioms that, in many ways, simply confound all our societal traditions and taboos. If, on a bad day, I inadvertently leave my back door open, that is not a license for you to pee in my toilet, empty my refrigerator, or install a wireless web cam. Even having burglar tools is a crime in most jurisdictions where there is anything worth stealing. How does this apply in the electronic sphere?

The idea of property that is most at issue is "exclusion," namely that if I have something you don't have it whereas if you liberate it from my dining room table you then have it whereas I no longer do. An MP3 of a popular tune is arguably like that, modulo a tendentious reading of contract and license, but certainly my private correspondence is something that you have no right to even if I fail to encrypt it to modern standards. The fact that an electronic property can be copied at virtually zero cost and yet with no exclusion to its holder is what makes this space hard.

USENIX did, when asked, take a stand on the publication of the Felten, et al., paper. As the entity with standing and with the authors credibly willing and able to share their information with our members, we did what we did. We did not, when asked, take a stand on the matter of Mr. Sklyarov. We will, doubtless, be asked to take positions on who owns what more and more since absent (God forbid) world government, there will always be jurisdictional diversity. If you haven't looked, the quantity of legislation filed on the issue of electronic ownership and affronts thereto is rising steeply. Law enforcement agencies at all levels are setting up electronic crimes task forces. Threat models now inform insurance rate setting and insurers will be adding a significant digit of precision to their estimates every time they can. My own business life (security consulting) is fundamentally built on maximing electronic property's value by minimizing its theft risk.

James Madison, in the Federalist papers, said that for democracy to survive it must avoid stable factions capable of imposing the tyranny of the majority. He said that the surest way to unstable factions is a differential ability to acquire property. As the percentage of the world's wealth that is information in electronic form grows, democracy will be front and center on protecting that property, or it will wither into tribalism. Lead, follow, or get out of the way?

The next time you hear that information wants to be free, or that some clever reverse-engineer has beaten some big, slow moving institution, think twice. Be careful what you wish for. The rabble's call to crush all forms of digital rights management in gladiatorial combat does not bode well for privacy because if information wants to be free, I'd like a copy of your genome by return mail.

Obituary: Jim Ellis

Our community recently lost one of its luminaries: Jim Ellis. Both Danny Smith and Jim Duncan share some words about his passing.

Danny Smith
danny.smith@sun.com

I worked with Jim Ellis for the last eight years in the FIRST community. Jim and I often exchanged technical thoughts and ideas on security problems that were current at the time, with the aim of providing solutions for people to limit the damage.

Jim and I performed a joint study to examine the potential threat of a multiplatform UNIX virus. We produced such startling results that we destroyed all of the material and ceased pursuing the examination. The risk of harm if those ideas leaked was just too great. Jim had great ideas! I am glad he worked on the defense side of the equation.

Jim's illness would wear him down. So he changed tactics: he rested when he needed rest, and he worked when he was able to work. He refused to let this illness stop him from doing the things he loved to do. He continued working full time until only shortly before he passed away. Even up to that point, he was producing excellent results and was heavily engaged in many projects. So successful was he, that many people either did not realize that he was ill at all, or did not realize the depth of his illness. His death came as a shock to many people within Sun as they were "only working with him just recently!"

Jim genuinely cared about people. He went out of his way to help anyone he could, and he never burdened people with his problems. Even as a manager, he would say, "I'll give you an update on how things are going, but you don't need to worry about this." He had an amazing ability to effect changes in projects without anyone losing credibility — he could engineer win-win situations in everything he did.

If I had a team of Jim Ellises, we could do amazing things. But, alas, there was only one, and he now leaves behind a legacy of what we should be doing. His courage, skills, and compassion were a true inspiration to us all, and he will be missed for a very long time.

Jim Duncan
jnduncan@cisco.com

Jim was the consummate scientist *and* gentleperson. There are others like him both in USENIX and the network security communities, but we need more.

At his funeral, as different folks talked about Jim (or "Jamie" as he was known to his family), it became obvious that his style carried over into lots of other things in his life. He and Carolyn were active in the home schooling movement, and he had the same profound yet subtle impact on the folks he met there. The same was true of their work with the League of Women Voters, and also with his family and friends, and his religion. Much of his extended family really didn't know about the details of his professional life, and they asked us geek types to fill them in with stories.

Through all of this, Carolyn and others insist that he wasn't afraid to die, and I believe it. I'm sure he wanted to find out all about it, and that attitude affected everybody around him. At the close of the church service, the minister announced that since the sky had cleared up again, we were all invited outside for the interment in the church cemetery. In true Pennsylvania fashion, the clouds abruptly opened up as the pallbearers laid down his casket, and we were all drenched. In cutting short his remarks at the graveside, the minister said that if he could, Jim would be explaining at that very moment the meteorological conditions that had caused the rain to fall.

Report from the USENIX Board of Directors

by Gale Berkowitz and Ellie Young

The following is a summary of some of the actions taken by the USENIX Board of Directors between January and August 2001.

The Board voted to allocate $50,000 for 2001, 2002, and 2003 to the Electronic Frontier Foundation (EFF) for legal costs associated with protecting copyright and fair use rights for DMCA legal cases. USENIX Board member John Gilmore and USENIX Executive Director Ellie Young were appointed the representatives of USENIX on the Felten, et al. lawsuit against the RIAA, et al. The Board also passed a resolution that USENIX will indemnify the Program Committee of the USENIX Security Symposium from any legal action that may be brought because of their decision to publish "Reading Between the Lines: Lessons from the SDMI Challenge".

The Board voted to allocate up to $21,000 to conduct a virtual classroom pilot project. The pilot project will assess whether or not the technology is sufficient, and gauge interest in the instructional modality. Proposals will also be sought for Web-based training and university program modules.

The Board voted to allocate $55,000 to the Software Patent Institute (SPI). These funds will be used for cleaning, formatting, and loading documents during 2001.

The Board voted to allocate $10,000 to the Richard Tapia Celebration of Minorities in Computing Symposium to be held October 18-20, 2001 in Houston. The funds are to be used to support students to attend the event.

The Board voted to allocate $5,000 to the Middleware 2001 Conference to be used for funding for students to attend the event.

Bylaws

A bylaws committee was constituted to review the USENIX bylaws and policies. The committee is comprised of Andrew Hume and John Gilmore from the USENIX Board of Directors, Attorney Dan Appelman, and Jane-Ellen Long from the USENIX staff.

Nominating Committee

Andrew Hume was voted to be Chair of the Nominating Committee for the USENIX Board of Directors.

Membership

Two special membership categories have been formed, one for retired persons, and another for persons with special circumstances that make the regular membership rate prohibitive. Membership fees for each of these categories is $50 per year.

LISA 2001 Conference Registration Fees

Conference registration fees for LISA 2001 will be increased by $15 to cover the costs of providing all registered attendees a copy of the forthcoming book *Selected Papers in System Administration* edited by Eric Anderson, Mark Burgess, and Alva Couch. The Board also voted to approve a $50.00 increase for both tutorial fees and technical session fees for those who do not use the Web to register for the conference.

Next Meeting

The next meeting is scheduled to coincide with the Annual Linux Showcase in Oakland, CA, on Wednesday, November 7, 2001.

2002 USENIX Nominating Committee

by Andrew Hume

Chair, Nominating Committee

The biennial elections of USENIX's Board of Directors will be held in the Spring of 2002.

Newly elected directors will take office at the conclusion of the first regularly scheduled meeting following the election, or on July 1st, 2002, whichever comes earlier.

There are eight board positions:

President, Vice President, Treasurer, Secretary,and four Directors at Large.

The new Board is normally a combination of current Board members and people new to the Board.

The vibrant health of USENIX and the technical strength of its offerings stems substantially from the vigor of its Board. Accordingly, there is a Nominating Committee whose charter is to present a strong slate of candidates for the election. (Candidates may also self nominate.) We are soliciting suggestions for nominees who are enthusiastic, energetic, responsible, and able to donate an appreciable amount of time to USENIX. Warning: this is a working Board, and not a resume stuffer; while not onerous, there is work to be done and failure to deliver will not only reflect poorly on the individual, but also negatively impact USENIX as a whole.

Primarily, candidates should have a strong interest in USENIX and its activities. Vision, passion, and the ability to work and play well with others are necessary. A sense of politics and management experience are increasingly important assets. However, the paramount requirement is the desire to make a difference and achieve something.

Please send suggestions for nominees (you can suggest yourself) by October 29 to: nominate@usenix.org

We also invite feedback (which will be kept strictly confidential) on the current board members.

Update on ReX, the International Research Exchange Programme

ReX, the international research exchange program co-sponsored by USENIX and Stichting Nlnet of the Netherlands, has recently funded four projects:

Delft University of Technology and Berkeley Wireless Research Center (UC Berkeley) to develop distributed localization algorithms for wireless sensor networks.

Tilburg University, The Netherlands and the Natural Language and Information Processing (NLIP) Group at the Computer Laboratory, University of Cambridge, UK to conduct research on the automatic construction of electronic dictionaries for use in text mining and related applications using memory-based learning techniques.

Universita' dell' Aquila, Italy and the Department of Computer Science at the University of Colorado in Boulder, USA, to develop novel wireless applications that leverage the Internet-scale publish/subscribe middleware framework of Siena.

The Cryptographic Group of Applied Statistical Unit in Indian Statistical Institute, Calcutta, India and the Department of Information Technology at Lund University, Sweden, to develop software oriented stream cipher for secure communication over network.

Funding still remains for 2001. The submission deadline for ReX proposals is November 1, 2001. For more information about ReX, please see: *http://www.usenix.org/about/rex.html.*

Announcement and Call for Papers

SANE 2002

3rd International SANE Conference

May 27-31, 2002

Maastricht, The Netherlands

A conference organized by the NLUUG, the UNIX User Group - The Netherlands
co-sponsored by USENIX and NLnet Foundation

OVERVIEW

Technology is advancing, the systems administration profession is changing rapidly, and you have to master new skills to keep apace. At the 3rd International SANE (System Administration and Networking) technical conference and tutorial tracks you'll find a wealth of opportunities to meet other system administrators and network (security) professionals with similar interests, while attending a program that brings you the latest in tools, techniques, security and networking. You can learn from tutorials, refereed papers and invited talks. Visit the Vendor Exhibition for the hottest products and the latest books available. The official language at the conference will be English. The conference will be located at the Maastricht Exposition and Conference Center, MECC.

IMPORTANT DATES

Extended abstracts due:	**October 1, 2001**
Notification to speakers:	**November 1, 2001**
Final papers due:	**March 29, 2002**

TUTORIAL PROGRAM - *May 27-29*

On Monday, Tuesday and Wednesday, a large selection of practical, problem-solving, in-depth tutorials will be presented to you by the most authoritative, popular and widely acclaimed speakers in the field. If you're interested in presenting a tutorial or would like to share ideas about what would make a terrific tutorial, please contact the Tutorial Coordinator by e-mail to <sane2002-tut-chair@nluug.nl>

TECHNICAL CONFERENCE - *May 30-31*

Thursday and Friday will offer comprehensive technical sessions, including keynote address, presentations of refereed papers and invited talks. Join peers and gurus during the enjoyable social event and the dazzling inSANE Quiz.

The SANE 2002 conference seeks original and innovative papers about the applications, architecture, implementation, performance and security of modern computing systems and IP networks. Papers that analyze problem areas and draw important conclusions from practical experience are especially welcome. Presentations are being solicited in areas including but not limited to:

- Security tools and techniques: IPSEC, Network Intrusion Detection Systems, Firewalls, VPNs, practical cryptography, auditing and computer forensics
- Attacks against networks and machines, including denial-of-service attacks
- Adventures in nomadic and wireless computing
- Web security fundamentals and practical web site maintenance
- Integrating new networking technologies like IPv6
- Network monitoring and traffic shaping solutions
- System and network performance tuning
- Managing enterprise-wide email and fighting SPAM
- Innovative system administration tools and techniques
- Distributed or automated system administration

REFEREED PAPER SUBMISSIONS

Papers for the technical sessions will be reviewed by the program committee. An award will be given at the conference for the best paper in this track. An extended abstract is required for the paper selection process. Abstracts must be submitted through the web form: `http://www.nluug.nl/cgi-bin/sane2002-abstract`. Abstracts accompanied by non-disclosure agreement forms are not acceptable and will be returned unread.

Authors of accepted submissions must provide a final paper for publication in the conference proceedings. These final papers are held in the highest confidence prior to publication in the conference proceedings. By agreeing to present your paper at SANE 2002, you also give license to the SANE 2002 conference organizers that it may be published on the NLUUG web site.

CONFERENCE ORGANIZERS

Program chair <sane2002-prog-chair@nluug.nl>
Edwin Kremer, TUNIX Open System Consultants, Nijmegen, NL

Tutorial Coordinator <sane2002-tut-chair@nluug.nl>
Jos Alsters, C&CZ, KU Nijmegen, NL

Program Committee
Jaap Akkerhuis, SIDN, Arnhem, NL
Walter Belgers, AT Computing, Nijmegen, NL
Ate Brink, Department of Computer Science, Utrecht University, NL
Rudi van Drunen, Leiden Cytology and Pathology Lab, NL
Peter Honeyman, CITI, University of Michigan, Ann Arbor, MI, USA
Brad Knowles, Brussels, Belgium
Brenda Langedijk, ITSX, Amsterdam, NL
Alexios Zavras, Lucent Technologies - Bell Labs, Athens, Greece
Kristijan Zimmer, FER / HrOpen, Zagreb, Republic of Croatia

Event Organization <sane2002-org@nluug.nl>
Jack Jansen, project coordinator, Oratrix, Amsterdam, NL
Wytze van der Raay, treasurer, NLnet Foundation, NL
MarÎlle Klatten, conference organizer, ICONIQ, Amsterdam, NL

Complete program and registration information will be available in December 2001. For the latest information about the conference, please visit the SANE 2002 web site: `http://www.nluug.nl/sane/`

For questions not being answered at this web site, please contact the ICONIQ office by e-mail: <sane2002-info@iconiq.nl>

CONNECT WITH USENIX & SAGE

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: +1 510 528 8649
FAX: +1 510 548 5738
Email: <office@usenix.org>
<login@usenix.org>
<conference@usenix.org>

WEB SITES

<http://www.usenix.org>

<http://www.sage.org>

EMAIL

<login@usenix.org>

COMMENTS? SUGGESTIONS?

Send email to <ah@usenix.org>

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to *<login@usenix.org>* or through the postal system to the Association office.

The Association reserves the right to edit submitted material. Any reproduction of this magazine in part or in its entirety requires the permission of the Association and the author(s).

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

12

;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