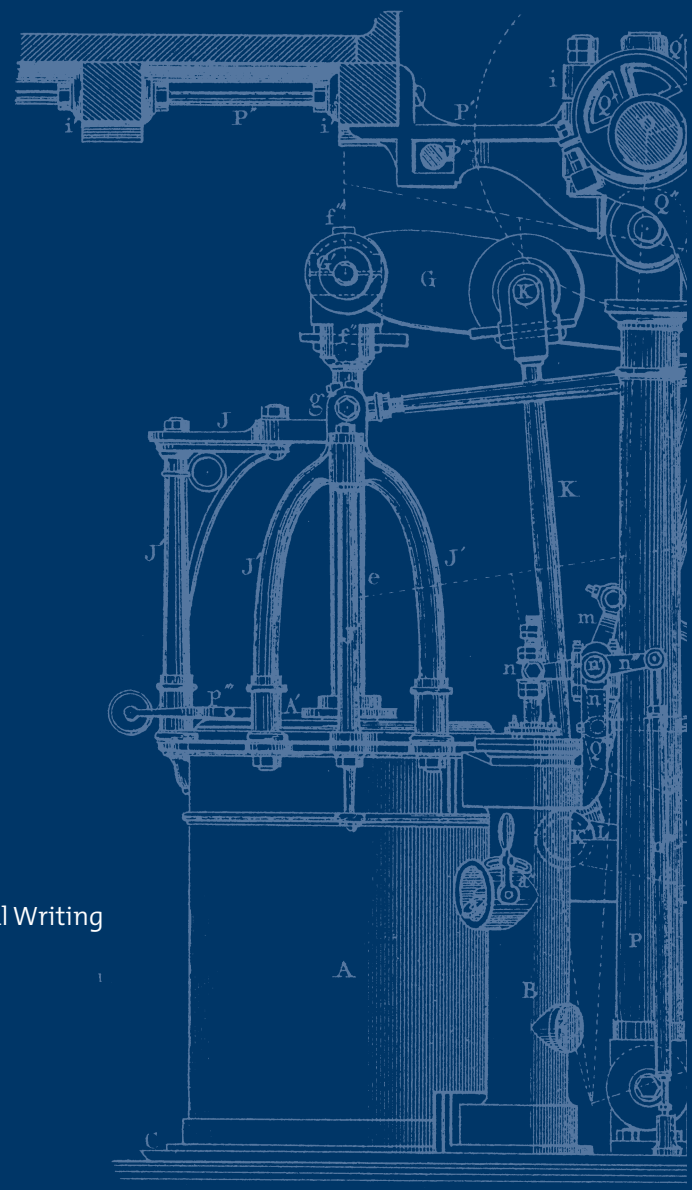
FEBRUARY 2005 VOLUME 30 NUMBER 1

;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events

2005 USENIX Annual Technical Conference

April 10–15, 2005, Anaheim, CA, USA
http://www.usenix.org/usenix05

2nd Symposium on Networked Systems Design and Implementation (NSDI '05)

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

May 2–4, 2005, Boston, MA, USA
http://www.usenix.org/nsdi05

3rd International Conference on Mobile Systems, Applications, and Services (MobiSys '05)

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

June 6–8, 2005, Seattle, WA, USA
http://www.usenix.org/mobisys05

First ACM/USENIX International Conference on Virtual Execution Environments (VEE '05)

Sponsored by ACM SIGPLAN and USENIX, in cooperation with ACM SIGOPS

June 11–12, 2005, Chicago, IL, USA
http://www.veeconference.org
Paper submissions due: February 18, 2005

10th Workshop on Hot Topics in Operating Systems (HotOS X)

June 12–15, 2005, Santa Fe, NM, USA
http://www.usenix.org/hotos05

Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI '05)

July 7–8, 2005, Cambridge, MA, USA
http://www.usenix.org/sruti05
Paper submissions due: March 30, 2005

Linux Kernel Summit '05

July 17–19, 2005, Ottawa, Ontario, Canada

14th USENIX Security Symposium (Security '05)

August 1–5, Baltimore, MD, USA
http://www.usenix.org/sec05
Paper submissions due: February 4, 2005

Internet Measurement Conference 2005 (IMC '05)

Sponsored by ACM SIGCOMM in cooperation with USENIX

October 19–21, 2005, New Orleans, LA, USA

19th Large Installation System Administration Conference (LISA '05)

Sponsored by USENIX and SAGE

December 4–9, 2005, San Diego, CA, USA
http://www.usenix.org/lisa05
Draft papers and extended abstracts due: May 10, 2005

4th USENIX Conference on File and Storage Technologies (FAST '05)

December 14–16, 2005, San Francisco, CA, USA
http://www.usenix.org/fast05

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

contents

;login:
VOL. 30, #1, FEBRUARY 2005

EDITOR
Rob Kolstad
rob@usenix.org

CONTRIBUTING EDITOR
Tina Darmohray
tmd@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
Steve Gilmartin
proofshop@usenix.org

PROOFREADER
proofshop
proofshop@usenix.org

TYPESETTER
Star Type
startype@comcast.net

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

office@usenix.org
login@usenix.org
conference@usenix.org

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

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

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

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

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

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

ROB KOLSTAD

motd

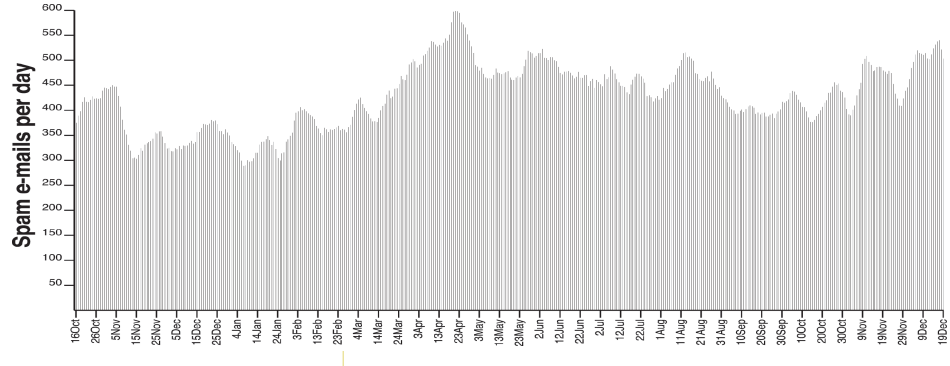
THE ONLY GOOD SPAM COMES FROM HORMEL

Dr. Rob Kolstad has long served as editor of *;login:*. He is SAGE's Executive Director, and also head coach of the USENIX-sponsored USA Computing Olympiad.

- *kolstad@sage.org*

Let's review some of the latest news about spam.

- Message security firm MX Logic reported that November compliance with the USA's CAN-SPAM act hit 6%, doubling October's 3% compliance rate. They add that 69% of spam is sent through "zombies," usually home computers controlled by spammers to send email on behalf of bulk mailers. In related news, a Maryland judge ruled that Maryland's anti-spam law is unconstitutional, since it "seeks to regulate commerce outside the state's borders."
- Iowa ISP Robert Kramer sued 300 spammers in an effort to stem the 10,000,000 daily spam emails he saw (in 2000). A U.S. District Court judge ruled that one spammer must pay him US$720M, and another must pay US$360M. Widespread opinion is less than optimistic about actually collecting these damages.
- Anti-spam firm Postini reports that 88% of email is now spam; 1.5% of those messages contain viruses. MessageLabs says as high as 6%, depending on the report. Every related news story I checked predicted that 2005 will see the true rise of phishing. (My own mailbox has seen no relief at all from spam. The graph below shows the rolling average of the number of my personal spam emails for the past 15 months.)



- The loss of productivity due to spam is gauged at anywhere from hundreds of dollars per year *per employee* on up. Administrators sometimes find themselves buried in spam or in requests to make it stop. Phishing is a US$137M–500M industry, depending on whose numbers you believe.

Now let's think back to the Golden Age of Email. You remember: each electronic message evoked thoughts of Christmas, like a package of joy waiting to be unwrapped. The bell rung by biff to signal a new message set off a Pavlovian salivation of anticipation at a new missive—perhaps it was a product order, a business prospect, or greetings from a long-lost friend. Those were the days!

Alas, those halcyon days of communication of a certain purity are gone. The tears have been shed; we've moved on.

What happened? In a nutshell, some lawyers in the Southwest tested the waters of Internet advertising and found them bountiful. Subsequently, every budding entrepreneur with a scam, fraud, herbal pharmaceutical, erotic Web site, or home-mortgage connection has decided that "almost-free" advertising can make big profits. It's almost as though someone is creating billboards that read, "If you can#t splel VaigrA, you cn mkae big MONEY wtih b.ul.k adtervising on teh I*N*T*R*E*N*E*T."

Some institutions are trying to stem the tide. We have black lists, white lists, and grey lists. We have black hole (non-)routing, sender protection frameworks, and, best of all, 150 vendors raking in two-thirds of a billion US dollars in 2004 just to slow the scourge. Venture capitalists pumped US$23M into anti-spam firms in August 2004 alone.

Current solutions seem to fall roughly into these categories:

- Stopping spam at the gateway to the local network (e.g., a list of unacceptable IP addresses)
- Filtering of spam using software (maybe by a third party)
- Laws that suggest spamming should be stopped
- Blocking by (a very few) ISPs of outgoing email connections from computers that seem to have been compromised

Perhaps looking at the bigger picture will help, since these measures are having an all-too-limited effect.

- Almost all spammers are motivated by money, although a tiny fraction are concerned with disseminating a political, a religious, or even a bizarre scientific message. The low up-front investment for universal and affordable access to the Internet (DSL, cable modems, businesses, hosting companies, even Internet cafes) drives down the entry cost. Crackers who take over others' computers (creating "zombies") send out more than two-thirds of the current flood of spam. Spammers would have no interest in this endeavor if they could not obtain customers. Behind-the-scenes reports reveal that spammers profit as do all scam artists: by selling dreams, appealing to greed, and selling items widely perceived to be unavailable in mainstream markets.
- Spammers enjoy anonymity. There are no means to complain about or avoid spam; requests to be removed from spammers' lists are widely believed to be ignored or, worse, are used as confirmation of the address.
- Spammers obtain their revenue at the credit-card-processing bureau, but the connections among their emails, credit card accounts, and true identities are not discernible by mortals.

Stopping any of a spammer's three enablers will thwart them:

- Remove access to cheap and easy sending of bulk emails.
- Remove anonymity: make spammers stand up personally for their products and services.
- Remove their ability to collect money easily and secretly.

I initially thought removing anonymity would solve the problem. Creating a positive identification token for tens of millions of Internet users is a very pricy proposition, and one not likely to serve the purpose. It appears that many people would gladly sell spammers their token for relatively small amounts of money.

Removing the other two enablers might yield better results. ISPs can either completely block outward port 25 traffic or restrict it to a set of well-known email servers. One would think it would be in the ISPs' best interests to shut down spammers. It's worked for Comcast, with 5.7 million subscribers. They implemented exactly this idea in June, stopping about 700 million emails per day.

Removing the ability to collect money is a very simple step to slow spammers: Identify a spam offer as fraud by purchasing the product and confirming that it is fraud, then (presumably with legal backing) work backward through the credit-card folks to shut down the offender. This seems like just the thing for our U.S. Federal Trade Commission. Existing legislation gives them plenty of ability to prosecute those who break laws not just once but millions of times. If the laws do not enable this, the laws need to be fixed. The "mood of the people" is such that this should work out quite easily. In fact, the state of Virginia sentenced a spammer to prison for nine years (though the constitutionality of that law is probably under study now as well).

Note that these proposals stop the problem at its source, not after it has consumed network bandwidth (not free), passed filters (not free; the manpower to deploy them has a cost), or even made it all the way to inboxes (where "just press delete" is a stupidity no longer even amusing).

The filtering folks, the black-hole list maintainers, commercial firms, and an heroic set of thousands of administrators are doing a great job of slowing the infection of this parasite on the Internet. None of them, however, has the ability to stop the problem at its source. ISPs and federal agencies do—and in many countries.

Why is spam OK? Why do we have to "take it"? I think we should do a much better job of encouraging those who *can* stop spam to do so.

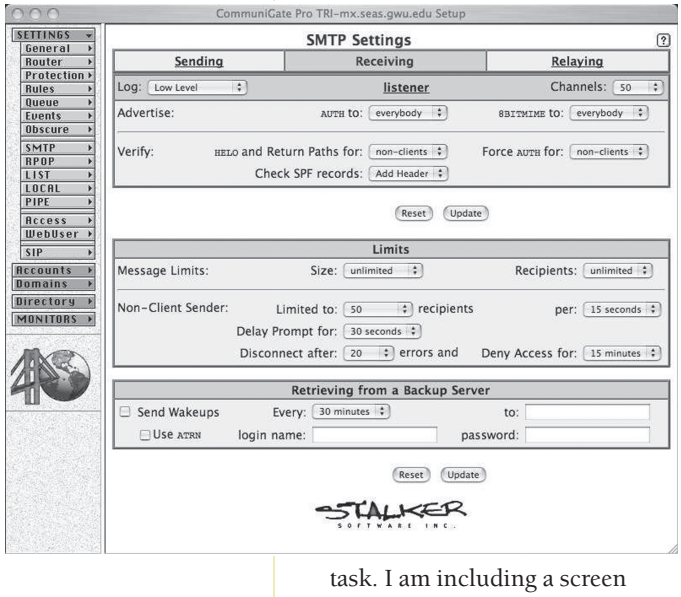
letters to the editor

TO: RHASKINS@CNETWORK.COM

Robert,

I just had the chance to read your response to Nick Christenson's letter to the editor in [the October] *;login:*. In your re-sponse you wrote, "I will admit, I haven't set up an MTA for SPF enforcement. But from a quick perusal of instructions for doing so, it does not look like an easy, straightforward task."

You should have written which MTA instructions you were looking at. SPF support is available in Stalker's CommuniGate Pro and is a straightforward task. I am including a screen shot to show that.



In the image you can see an option to specify checking SPF records during SMTP receiving, where the options are: disabled; add header enabled (reject on fail).

I understand that your purpose was to introduce SPF, but I'm writing just to say that I tend to agree with Nick. I'm certain that more MTAs can be made to use SPF quite simply, so there would be no artificial hurdle to adoption of the technique. Based on my experience so far, the biggest hurdle is changing MTAs so that forwarding doesn't break SPF (but I've found that other techniques, such as external POP polling, can be used to circumvent the problems with forwarding).

Kindest regards,

JASON MADER
jason@ncac.gwu.edu

ROBERT HASKINS REPLIES:

I was talking about integrating SPF with Sendmail. It looks as though it's easy to integrate SPF enforcement into Communigate Pro!

DEAR EDITOR:

In "Wireless Security: A Discussion" in the December *;login:*, Marcus Ranum raised a number of valid points, but I was most struck by his complaint, "I was going to be forced to waste time installing security measures." Wow. Isn't this exactly the "luser" attitude many of us fight every day? Hasn't making and selling these "security measures" been a large part of Marcus's professional life? If even he is seriously annoyed at the prospect of actually having to use security measures, we must have a terribly long way to go.

If I may be permitted an analogy: If I don't want people peering at the cargo in the back of my vehicle parked at the conference hotel, I put the cover over it, rather than bitch at the guy who says, "Hey, I can see your laptop in there!" (Nor do I run to "mommy": "He's peeking, he's peeking!").

JOHN HASCALL
Iowa State University

MARCUS J. RANUM

who needs an enemy when you can divide and conquer yourself?

Marcus Ranum is CSO for Tenable Network Security and a well-known security products designer and implementer. He holds the TISC "Clue" and ISSA Lifetime Achievement awards and is the author of the recently published *The Myth of Homeland Security* (reviewed by The Bookworm in this issue).

mjr@ranum.com

I SURVIVED THE UNIX WARS, UNLIKE most of the companies involved in them. Perhaps you remember Pyramid, SCO, Apollo, DEC, Sun, Silicon Graphics, Gould, and so on. In their day, they fought ferocious scorched-earth wars trying to win customers' minds and money. The survivors, with the exception of Sun (a.k.a. "The Last Man Standing") have either gone into the mists of time or are niche players forced into new markets in order to survive. Other than their conflict, what did they have in common? They were all selling some kind of UNIX operating system.

Back in the UNIX wars, the vendors had two primary axes on which they could compete: hardware speed and features of their UNIX flavor. Toward the end of the UNIX wars, a third battle evolved, surrounding the "desktop metaphor"—the look and feel of the workstations' GUI. If you were around back then, you'll remember the ferocious fights over whether or not the "3D-look" widgets of the Open Software Foundation (OSF) Motif metaphor were just flash and glitter or whether they were actually kind of cool. Today, few remember the argument, and the code in question would be considered remarkably tight and lightweight compared to what people now use. If you step back and look at the UNIX wars from a high altitude, the actual battlefield was very small—GUIs and features in a UNIX operating system don't really sway customers much; the vendor who won (Sun) did so because they offered a consistent software experience (SunOS, later Solaris) across a broad spectrum of hardware at different performance levels from desktop to data center. In other words, the customers didn't care if the GUI had a 3D look and feel as long as it was fast, reliable, and affordable.

You don't need to be an advanced student of history to know what happened. While the UNIX vendors beat each other up over what amounted to nitpicking details, another vendor offered the same consistent kind of software experience across a broad spectrum of hardware (including laptops)—I am referring, of course, to Microsoft/Intel. Through the lens of 20/20 hindsight, it is clear that the UNIX vendors were shortsighted losers arguing over what to watch on television and fighting for the remote control while the house burned down around them.

Now, read this carefully: I am not bashing Microsoft Windows. As a UNIX system administrator with 20+ years of experience, and a Windows system

administrator since Windows 1.0, I can tell you that there isn't a whole lot of difference in the workload of efficiently running either environment. Sure, there are lots of annoying details in either one, but it takes about the same time for an expert to load and configure each system. (In the old days, UNIX machines were faster to bring online because of the prevalence of decent tape drives, while Windows was primarily loaded by floppy, but that was about the only difference.)

In other words, customers didn't "choose" Windows because it was better (or worse) than UNIX—they did it because Microsoft/Intel was careful to guarantee them a consistent software experience across a broad spectrum of hardware. And, of course, the application developers flocked to that consistent software experience because it meant their products were cheaper to develop without the headaches of version-specific differences.

In 1985, when I wrote code for my UNIX machine, it worked on all the other UNIX machines because there was basically a single flavor of UNIX, which all used the same compiler, and everything just worked. Today, you actually have to be quite careful if you want to write code that compiles and works correctly on Solaris, Linux, and BSD.

Indeed, most "open source" packages now include special tools that dynamically reconfigure the code based on complex knowledge bases that encode the differences in how Solaris says "tomato" and Linux says "tomahto." It takes longer to configure code than to compile it, these days, which is categorically not the case on Windows. Windows stuff just works and usually keeps working. Do you think that this might, just maybe, have something to do with why major apps like Adobe Photoshop, Macromedia Director, Adobe Premiere, etc., are still not available on UNIX and *never* will be?

Why is all this relevant? Because the UNIX wars didn't end and, consequently, the "last man standing" is still Microsoft/Intel.

What do I mean, "They didn't end"?

I installed Linux on one of my systems the other day so I could use it as a teaching vehicle for my class on system log analysis. But first I had to email a bunch of my friends and ask them, "What version of Linux should I use? Red Hat? Debian? Gentoo? Mandrake? Slackware? Do you think I could get away with OpenBSD or FreeBSD?" The responses I got indicated that none of my friends use the same thing but that I could be sure that if I used Flavor X some adherent of Flavor Y was going to bust my chops about it, and that someone was sure to show up with Flavor Z and have trouble making things work. Do you hear the sound of distant laughter coming from Redmond? I do.

The early days of the Linux movement were heralded with grand pronouncements of war to the death with Microsoft—war from the desktop to the data center, and a free, compatible high-performance alternative to Windows. What I see now is that the open source movement was more like a 14-year-old punk standing in the street telling Mike Tyson that he had an ass-whipping coming. Not the Mike Tyson we see today, either, but the Mike Tyson who could deliver a line-straight punch that could knock a hole through the side of a steel I-beam.

Unlike Tyson, no doubt, Microsoft was at least courteous enough to pay lip service to the threat that the 14-year-old was making, using Linux as a "credible threat" to help argue that Windows was not, in fact, a monopoly. Guys, let's face the facts: Windows is a monopoly because short-sighted open source geeks and UNIX weenies were too busy squabbling over whether RPM was better than build-from-source or Gnome versus KDE, etc., ad nauseam.

The tragedy here is that, unlike during the UNIX wars, the battlefield now is even more narrow. The hardware spectrum is a constant, so system performance is barely an issue: Nobody measures whether Slackware is faster than OpenBSD, and if someone did, nobody'd care anyhow. So the battle in the free UNIX space is entirely over command line options, system administration paradigms, installation packaging, and 3D GUI features.

I've got news for you: Real Programmers Don't Care about that garbage. Has it managed to completely escape the attention of the open source movement that Adobe, Macromedia, Corel (mostly), and so forth have blithely continued to be non-UNIX while waiting for the dust to settle? Only now they have realized that it won't settle and that oh-so-quietly the rush of announcements of support for Linux has not translated into a rush of quality applications.

Let me make a prediction for you. The open source movement is not going to hurt Microsoft to any significant degree. But it'll put Sun out of business. Good move, guys! Do I hear the sound of distant laughter from Redmond?

Is it too late to save the situation? Yes, I think it is. At this point, there are too many adherents of, and too much investment in, the "not invented here syndrome" for anybody involved in the various free UNIX flavors to come to their senses until there is only one man left standing.

But that's not going to happen because, with free software, it doesn't cost very much to remain standing forever. It's an issue of ego, not technology, so don't expect sense or sanity to kick in. We all know the expression "divide and conquer," but Microsoft didn't even need to do that—they could just sit back and watch free UNIX

fail to become a credible threat because, well, frankly, it was in the hands of egotistical detail-oriented amateurs.

Who's left that can compete with Microsoft? The place to look for alternatives is wherever there is a broad spectrum of hardware with a consistent software experience. That doesn't leave much: proprietary devices like PDAs and gaming consoles. If you want a high-impact platform that doesn't come from Redmond, look to what the grown-ups at Sony are producing for their Playstation 2 network. The platforms are consistent and won't fragment into competing versions, because they are proprietary and the folks producing them are in business to make money, not for their personal gratification and lust for limelight. Or if you want a consistent software experience—go with Windows.

Remember: Real Programmers Don't Care.

ROB KOLSTAD

SAGE–news article on SCO defacement

kolstad@usenix.org

EACH WEEK OR SO, A SUMMARY OF 6–20 Web articles of interest to the system administration community is sent to the sage-news@sage.org mailing list, which has about 1,200 members.

On Thursday, December 2, the following article appeared:

> #######
>
> Vandalism: SCO site vandalized
>
> news.com has a screen shot of the SCO website hack. It's interesting to see what unethical people can do to a company presumably pursuing lawful royalties for its intellectual property.
>
> [I found it amusing…RK]
>
> http://news.com.com/Image+Screen+shot+of+SCO+hack/2009-7349_3-5469293.html

Soon thereafter, Steven Jenkins and I had an exchange on the roles and duties of editors in situations like this. This column summarizes that discussion and attempts to clarify the issues raised and resolutions we came to.

The email exchange will be presented here as a one-on-one discussion, in the belief that a discussion format is much easier to follow.

OUR DISCUSSION

Steven: I don't think that SAGE should encourage this type of illegal behavior. The second sentence in the summary just didn't discourage hacking as much as it should have. Furthermore, I read the editorial comment as being supportive of this sort of activity. I realize that humor is very difficult to convey in email (or written word). However, given the comment's terseness, it can certainly be interpreted as SAGE finding pleasure in the defacement of SCO's property.

Rob: I can see how the bracketed editorial remark's antecedent for "it" isn't clear. I meant the image itself was amusing. I imagine the word "interesting" was potentially not as well chosen as it could have been. Did you find the image amusing?

Steven: Yes, I personally found the image funny. But I don't think it's appropriate for the professional system administrators organization itself to make light of the illegal defacing of a Web site. Keep in mind that system administrators at SCO will have to replace the original Web site, track down how the vandals broke in, and then address those security holes. SAGE represents those system administrators!

Rob: Ignoring the actual behavior that triggered this apparent defacement, are you saying that my editorial comment (which was bracketed and signed) would still, even with a properly expressed antecedent, be out of line?

Steven: I think that organizations and the editors of publications should be held to a higher standard than a random organization member. I'd be somewhat embarrassed if a SAGE member publicly cheers about this; it's worse when an editor cheers.

Rob: Of course, I'm not representing SAGE in bracketed, signed remarks. I really didn't mean to encourage such things either. The news, however, was interesting in the context of our community.

I would be sorry if my amusement (which is apparently shared by you) were to be taken as encouragement of such activity. But it seems that the raising of the bar or standard you advocate would require me to refrain from comments of all kinds, and that doesn't feel right, either.

Steven: Editorial pages (or remarks) are taken by many to be the view of the publication, unless there is a stable of known editors with varying viewpoints. I think the same holds true for electronic newsletters.

OUR CONCLUSIONS

After a telephone conversation between Rob and Steven, a few points were agreed upon:

- The newsletter footer will explain that editorial comments represent the editors' opinions and not SAGE's.
- We will expand the editorial board, most likely to include Steven.

If you're interested in joining the editorial board, we're always open to wider participation! Contact sage-news-owner@sage.org to join the team.

HEISON CHAK

voice over IP with Asterisk

Heison Chak is a system and network administrator at SOMA Networks. He focuses on network management and performance analysis of data and voice networks. Heison has been an active member of the Asterisk community since 2003 and will be delivering a tutorial on VoIP Principles and Practice at the USENIX '05 Annual Technical Conference.

heison@chak.ca

ASTERISK IS A SOFTWARE-BASED PBX that runs under Linux, *BSDs, and Mac OS X. It can also be used as a VoIP gateway and can provide a bridge to the PSTN (Public Switched Telephone Network, a.k.a. Plain Old Telephone Service). IAX (Inter-Asterisk eXchange), pronounced "eeks," is a protocol used by Asterisk as an alternative to SIP, H.323, etc., when connecting to other VoIP devices that support IAX.

This article describes some of the interesting things I have done using Asterisk and VoIP, told as the story of a business trip from my home in Toronto.

It was 12:15 in the morning and I had just gotten home from work. Realizing that I had not yet packed for my trip to San Francisco, I was a little worried about missing my early morning flight. I knew that snooze alarms no longer worked for me; I would often keep hitting the snooze button when the alarm went off, or I would just be so used to the sound that I wouldn't even hear it at all. Either way, I could wind up being late. The AGI (Asterisk Gateway Interface) script I had installed on my Asterisk box that serves wake-up calls solved both my problems with snooze alarms.

Wake-Up Calls

Before going to bed, I dial ext. 100 from any phone in the house to request a wake-up call, and the voice of Allison Smith (Allison is the voice of Asterisk) prompts me for the desired time. At 6:30 a.m. that morning, Asterisk called the auto-answer extension of my IP phone in the bedroom. Besides telling me that it was a wake-up call and announcing the current weather in Toronto, Allison also challenged me to repeat a four-digit number after her. If I failed to respond or if I hung up after three tries, Asterisk would call for help, playing back my own recorded voice begging her to wake me up on that same auto-answer extension. The last resort would usually get me the unpleasant voice of my mother.

Dynamic Contents

That morning, I managed to get out of bed before Asterisk escalated the call, yet I missed the weather report when Allison did the announcement. I wanted to double-check the weather in both Toronto and San

Francisco, as well as make sure my flight was on time before heading out to the airport, but my laptop was already packed away.

With the help of a PHP script on my Web server, I was able to look at reformatted contents from theweathernetwork.com and aircanada.com on my IP phone as I hit the service button. Calling ext. 102 invoked weather.agi, allowing me to get a more detailed weather report based on the three-digit airport code (e.g., 736 for SFO and 999 for YYZ). A TTS (Text-to-Speech) engine from Cepstral was used to announce the weather forecast of the requested city. After hearing that Toronto's temperature was in the minus 40s, I was happy to find out San Francisco would be in the mid-60s throughout the week.

IAXy

Given it was a Monday morning, the check-in line-up wasn't so bad. However, I was not making friends with airport security, especially when they found this little blue thingie in my luggage and wanted me to explain what it was: "Okay, this blue thingie is called IAXy (pronounced 'eek-see'); it's an analog telephone adapter that allows telephone calls to be made over the Internet with a regular telephone, and the 'y' in IAXy doesn't stand for anything, it's just a product from Digium (*http://www.digium.com*) with a cool name."

Then, I heard a very familiar voice saying, "Yeah, this is similar to what Vonage and other voice-over-broadband providers are offering, but he is doing it with some open source software." I turned around and there was my boss, who was traveling with me. I was glad he was there, or there would have been a good chance I'd have missed my flight.

Toll-Free Service

I promised to call home when I landed, but my cell phone had very poor reception at the baggage claim. So I went to a pay phone and dialed my home number without inserting any money. . . and it worked! That was my first time calling my toll-free number—my VoIP provider was charging $0.03/min. While most VoIP providers charged a monthly access fee plus usage, I chose one that supported IAX on a prepaid account with no monthly fee. Calls made within US48 were $0.03/min, while calls made within Canada were only $0.02/min. My VoIP provider also provided a Web page for customers to monitor their usage.

After the call had terminated on the PSTN in Chicago, it was delivered to me via IP over the Internet to my Asterisk server behind a firewall. The voice on IVR (Interactive Voice Response) sounded very much the way it does on a landline.

I entered my wife's extension number, and the IVR responded with, "Please wait while I connect your call." My wife and I talked for a few minutes before hanging up. Next, I tested the calling card extension and entered my boss's mobile number. It freaked him out when he saw my home number showing up on his mobile when I was standing right next to him.

Telemarketers and Junk Faxes

As soon as I had network connectivity for my laptop, I wanted to check my email and there were a few .wav voicemail attachments from the PBX. Upon receiving voice messages, Asterisk would send a notification to email with an attachment of the voicemail file. Clicking on the attachments allowed me to listen to the recording without dialing into the PBX. I was also interested in other calls that I missed during the flight, and that's when my PHP-based CDR (Call Detail Record) report came in handy. Through Mozilla, I queried against my

PostgreSQL-based CDR and saw multiple entries of the same toll-free number. Taking a closer look at the CDR, I suspected it was a telemarketing firm who had made many attempts in the past few days but kept getting trapped in the IVR menu until the telemarketing agent finally hung up.

Blacklisting such a telephone number would result in a fast busy signal by Asterisk upon receiving CID (Caller ID) matched calls. In most cases, my number was removed from their database within two more attempts. Incoming faxes could also be blocked with this technique, after identifying the CID from which the junk originated.

Free Long Distance

Most hotels in the Bay Area offered complimentary high-speed Internet connectivity, and I was looking forward to making free long distance calls over IP. I connected my IAXy to the RJ45 outlet and moved the telephone in my room over to the IAXy from the wall jack. Unfortunately, there was no dial tone when I picked up the handset.

To verify that my problem was not related to the IAXy, I connected my laptop to the network and soon found out that I needed to register to the hotel's proxy before using the high-speed connection. Knowing that I could not use the IAXy, I started up iaxcomm (an IAX-based soft IP phone, http://iaxclient.sourceforge.net/iaxcomm/) in Linux and used the microphone and speaker on my laptop to chat with my wife. Unlike the toll-free calls I made earlier that cost me $0.03/min, calls made with iaxcomm were completely free, even if I were to call someone in Toronto with iaxcomm, since Asterisk was providing a bridge to the PSTN. During the conversation, we noticed some acoustic echo issues, but these were resolved when I switched over to a headset with a built-in microphone.

Dialplan

To demonstrate how Asterisk can be used to filter unwanted numbers (a.k.a. "the 'ex-girlfriend' feature"), let's examine the dialplan, stored in the file extensions.conf. The Asterisk dialplan file consists of a collection of contexts. Each context consists of a collection of extensions. The priority of an extension specifies the order in which it is executed by Asterisk.

```
[context]
exten => <extension>,<priority>,<application>(<args>)
```

In the following example there are two contexts, [incoming] and [real_extensions]. Extensions 100, 101, and 102 have been included by the [incoming] context using the include command.

When the [incoming] context answers a call with extension "s," it executes each line of exten => s, according to the priority. It performs Caller ID number matching. If there is a match, it plays a fast busy tone. If all matches fail, it moves on to priority 30, plays the IVR greeting, and waits for a number to be entered. If a caller enters 101, it will then ring my IP phone for 20 seconds before passing the caller to the voicemail application with the unavailable flag, "u."

```
[incoming]
exten => s,1,Wait,1                ; Wait a little for CallerID
exten => s,2,Answer                ; Asterisk answers the line
exten => s,3,DigitTimeout,5        ; Set Digit Timeout to 5sec.
exten => s,4,ResponseTimeout,10    ; Set Response Timeout to 5sec.
;
; Black listing numbers
```

```
;
; Jump to priority 20 if CID matches, else goto priority 6
exten => s,5,GotoIf($[${CALLERIDNUM} =  18668493243]?20:6)
; Jump to priority 20 if first 6 digits of CID match, else goto priority 30
exten => s,6,GotoIf($[${CALLERIDNUM:0:6} =  647722]?20:30)
;
; Banned numbers
;
exten => s,20,Congestion    ; Plays the fast busy tone
;
; Auto attendant says,
; "Thank you for calling, if you know the 4 digit extension..."
;
exten => s,30,BackGround(ivr-greeting)
include => real_extensions
[real_extensions]
exten => 100,1,AGI,wakeup.agi
exten => 101,1,Dial(SIP/cisco1,20,Tr)
exten => 101,2,Voicemail(u101)
exten => 102,1,AGI,weather-station.agi
```

Conclusion

I've described a few of the interesting ideas that can be accomplished with Asterisk and some programming. Most of these ideas are based on the Asterisk dialplan and AGI (Asterisk Gateway Interface). While the dialplan handles routing of calls through Asterisk, the AGI provides hinges for the logic to extend beyond the dialplan with other programming languages. For rapid development/deployment, most people use Perl and PHP to write AGIs. However, if you are concerned about performance, you may want to execute only AGIs that are written in C.

STEVEN ALEXANDER

why teenagers hack: a personal memoir

Steven is a programmer for Merced College. He manages the college's intrusion detection system.

alexander.s@mccd.edu

BROWSING THROUGH SOME OLD issues of *;login:* online, I came across the "Point/Counterpoint" that Tina Darmohray and Rob Kolstad wrote in April of 2002 about the possible culpability of the security community in the hacking career of Ben Breuninger, a.k.a. konceptor. (The young man was invited to represent the "black hats" at a computer conference in Orlando, Florida.) Both their points of view are interesting. I do agree with Rob that a twenty-one-year-old ought to have known better. But sometimes growing up is hard to do.

I don't think anyone intended the 1998 conference to be exploitative, but I can imagine the unintended consequences it may have had. While the "good" attendees may have encouraged Ben to pursue computers in other ways, being funded to appear at the conference must have stroked his ego fiercely.

I understand this because, unfortunately, I myself did not start out on the right side. I straightened myself out a long time ago, but for a while things were a little different from the way they are now.

The dark side was very tempting because it offered numerous benefits, both perceived and real, for a teenager. It was rebellious and cool. It was empowering to know more about computers and networks than the adults around me; break-ins were a way of proving this. Nobody my age cared that I could write neat programs in assembly language or even cared to know what the hell assembly language was.

Quake skills would win you a few friends, but my computer couldn't run the game, and I wasn't particularly interested in playing video games anyway. Hacking?[1] That was cool. Almost none of my peers understood anything about the mechanics of it—what they did understand was picked up from movies like *Wargames* and (shudder) *Hackers*—but damned if I wasn't sticking a finger up at the authorities and the system.

One of the worst aspects was the wild myths that surrounded hacking. I once heard the insane assertion that anyone who was able to hack MIT's network would be admitted into the MIT CS program. Stories of companies hiring hackers for six-figure salaries floated around here and there. Most unfortunately, there is truth to some of the stories.

I think that it is important for adults to reach out to young black hats and try to convince them to pursue their interest in security through other means. One of

my high school teachers was instrumental in bringing me around. Sadly, many adults are quite unhelpful.

I became interested in computers at 13, started hacking by 14, and gave hacking up at 16; I had finally grown up enough to join the mainstream. I had skated through my first two years of high school not trying to do particularly well. By the time I was a junior, I knew that I wanted to go to college (yay!), major in computer science, etc., and I knew what I needed to do to get there. The major reasons that I quit hacking are that I no longer valued the popularity that it brought me, I realized the consequences it might bring upon me, I came to understand the stress and problems it was causing admins worldwide, and I figured out that I wasn't really respected, just feared. In other words, I grew up.

The problem was that in about the second or third week of my sophomore year, I had hacked my high school's Web site. I got away with it—barely. My IP address was logged. The school contacted my ISP. The ISP thought it was me but said they couldn't swear to it (I don't know why; maybe the log times were sketchy). They knew, I knew they knew, and I knew they couldn't prove it.

Fast-forward a couple of years. I'm a high school senior. I'm assembling applications for various colleges. I'm pulling straight As, have strong SAT scores, AP classes, etc. Two Windows NT servers crash in the middle of a Service Pack 3 install and someone decides to blame me. I'm pulled out of class, searched, have a floppy disk confiscated, and am interrogated for three hours by three vice principals. The computer I use at school is searched, but the only "bad" thing on it is mIRC.

My stepdad and I met with three vice principals, two network admins, and a police officer. One of the network admins explained that they knew that I had been responsible, because the IP address of the computer I used in my programming class was displayed on the trademark Blue Screen of Death when the servers crashed (no, I'm not kidding). I wasn't in the class with the suspect computer when the "attack" occurred, but the theory was that I had written a program to crash the servers at a certain time, then erase itself.

They never did settle on a technical explanation for the crash. The last I heard, they suspected it was a one-megabyte IP packet (I'm still not kidding, and, no, I have no idea who came up with the notion). Since I was officially in violation of the school's acceptable-use policy by having mIRC installed, I was banned from using any school computer. Since I couldn't use school computers, I was dropfailed from my programming class and my networking class. My stepdad bought the administrators' reasoning, of course.

The networking class was a work-study class that had me working at a local ISP (my ISP, the one that almost caught me!). Someone (one of the vice principals, most likely) called the ISP and talked to my boss, the network admin. They explained everything and laid out the "evidence" for him. The school suggested that the ISP fire me. My boss, thank Joe Pesci,[2] told them basically to go to hell. As he put it, in rather unkind language, their admins were liars and idiots to boot.

My point is that, while some professionals try to help talented youths apply their skills in positive ways, other adults are not so reasoned in their approach.

These days I look at what happened as having been, for the most part, a karmic kick in the ass. The whole situation did, after all, start with me doing things I shouldn't have been doing. But at the time, I was not encouraged to move over to the right side. If it had not been for some very supportive people, primarily my boss and my girlfriend (who is now my wife), I don't think I would have changed. My gut reaction was to grab my assembly manuals and write an implacable, firmware-flashing bit of madness to wreak havoc throughout my school district. It was hard to walk away.

It is unbelievably empowering to know that you don't have to play by the same rules as everyone else. I understand from the "Point/Counterpoint" article that Ben Breuninger didn't have a grudge against LLNL. I don't think that mattered. Hacking gives you the ability to exist outside the law (for a while). The risk is relatively low, and you're aren't hurting other people in a way that is tangible, as it is with other crimes. (Of course, those in the DoS business are looking to cause tangible losses.)

I felt spurned by "the system" when I got into trouble. Before that, I felt empowered. I think others also feel empowered by the respect and awe they are handed and believe that they are placed above or outside the system because of it. Some feel that the system will actually reward their transgressions if they prove themselves skillful enough.

I don't think there is a lot that can be done once people who should have grown up already act out and find themselves on the wrong side of the law. I am very glad that I grew up before I met the fate others, like Ben, have suffered.

I think that hiring convicted or active hackers sets a terrible precedent. It reinforces the myths that make the dark side alluring. I don't think that learning from a teenage hacker or respecting his knowledge on technical matters is bad. I do think these kids should be encouraged to do more productive things (kudos to Rob on USACO in this context). Those who are caught young enough should be both punished and steered in new directions.

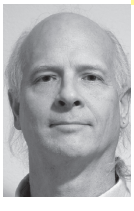
The bottom line is, however, that some kids will not grow up. They will continue to make poor choices and to break the law. They will continue to value the adulation of their peers and the myths of idiots over the well-reasoned advice of adults and professionals, and there really isn't a thing anybody can do about it.

1. Forgive me for using this in the media-driven rather than kernel-hacking sense.

2. See George Carlin's *You Are All Diseased*.

RIK FARROW

musings



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

A CURIOUS CONFLUENCE OF EVENTS has prompted me to renew my acquaintance with NFS security. For some reason, I had allowed NFS to slip from my awareness some time ago. Perhaps it was because the problems facing those who wanted to use NFS securely seemed overwhelming. Just as likely, other new things that glittered and shimmered with expectation grabbed my attention, leaving the venerable NFS to sit ignored in a corner.

But only by me. Many organizations rely on NFS, including my friends at San Diego Supercomputing Center. Elsewhere in this issue, you should find a tale told by Abe Singer about the experience he and his fellow workers had with an intrusion that began in Spring of 2004. The intruder continued to abuse SDSC systems for weeks and, at some point, took advantage of old, known weaknesses in NFS to do so.

While in Atlanta teaching my class for LISA, I asked how many people were using NFS in their organizations and was surprised by the result. I guess I had been asking the same question in the wrong venues, because lots of people indicated that they were using NFS. That, coupled with listening to Brian Pawlowski's (Netapp.com) Invited Talk about the future of NFS, really got me interested.

The Past

NFS started out at UC Berkeley and then was adopted by Sun Microsystems. The concept of a network file system didn't start with NFS but had (at least) one earlier, significant implementation. Apollo Domain had a network file system, along with single sign-on, unified user accounts, and home directories supported via the network file systems. You could log on to any Apollo workstation and get your home environment. While this may sound familiar, the underlying architecture paid serious attention to security. Apollo also had signed patches that could be centrally installed by the system administrator. But I digress.

NFS security, ever since its beginning, had barely existed. I won't say it didn't exist; that would be unfair. And today, most organizations continue to use the earliest form of NFS security, not because nothing else exists, but because it is the lowest common denominator and works on anything that supports NFS.

In the past, NFS security took two forms: user identification, and IP address–based access control. Let's talk about user identification first.

The user identification is called AUTH_UNIX, a somewhat misleading name, as AUTH hints at authentication, when there really is none at all. In AUTH_UNIX, each Remote Procedure Call (RPC) request sent to an NFS server includes the requesting user's ID number (UID). The UID gets assigned to your account via your /etc/passwd entry associated with your login shell and is used to signify ownership of files by being included in file attributes (within inodes). You can see the UID and GIDs (group IDs) associated with files when you use the command ls -ln (the -n option suppresses the conversion of UIDs to names).

NFS includes this very same UID, along with up to 16 GIDs, in each RPC request. The UID and GIDs will then be used to determine access to files and directories on the NFS server. Perhaps at this point you have discerned why I don't call AUTH_UNIX authentication. While a UID may be used to identify a user, it is not in itself a form of authentication. Anyone who can manipulate the RPC request can insert any UID he or she wants. No authentication is required to do this. None at all.

A Curious Tool

Back in 1991, Leendert van Doorn wrote a tool he called the nfsshell. The nfsshell generates legal NFS client requests, with the optional feature that the user of nfsshell can insert the desired UID into any request. No special privilege is required to use nfsshell—that is, you can run nfsshell as an ordinary user and can easily masquerade as other users when accessing an NFS server.

Van Doorn, who now works for IBM (he is shown acting as an antenna in his signature page, http://www.research.ibm.com/people/l/leendert/), created nfsshell to demonstrate the insecurity of AUTH_UNIX. You can find a copy of nfsshell on Van Doorn's Web page, although you will need to find patches, and perhaps an include file, if you want to get this working on Linux.

I configured an old Solaris system (2.7) to act as an NFS server, then tested it using a Linux system. Then I fired up nfsshell and started experimenting. Nfsshell worked as advertised in that I could create a file on the NFS server that could only be read by some other user, change the UID with nfsshell to be the same as that user, and read that file. The interface to nfsshell is similar to FTP or smbclient in general outline, and not at all hard to use.

One saving grace to NFS that I should point out is that by default UID zero, or root, gets mapped to -2 (username "nobody," by tradition). So changing the UID to 0 in nfsshell usually works worse than picking on some other UID. Note that you can disable this safety feature on your NFS server by exporting or sharing directories and using the option anon=0. You *really* don't want to do this. You can use the root option with a list of hostnames if you really want to permit root access to specific systems. This feature was designed for diskless workstations, pretty rare beasts these days.

At this point, NFS users should be either experiencing shock or muttering to themselves, "I knew it was so." On a more practical note, you should be wondering what you can do to add real security to NFS. After all, this is a disaster, isn't it?

NFS also started out with host-based access control. That is, the person who configured the NFS server could limit which clients could mount an exported ("shared" in Solaris) file system. There have been loads of problems with this scheme, some involving bugs, others involving configuration errors, and many meaning that file systems were exported to the world. Dan Farmer and Wietse Venema's SATAN scanner (1995) alarmed the world when released, as well as alarming NFS administrators when the tool would announce file systems exported to the world.

Another problem with earlier versions of NFS is that they relied on UDP for transport. UDP was chosen because it is stateless and was quite a bit faster than TCP back in the early eighties. UDP is also trivial to spoof, making it easy to get around the host-based access control, which relies on the IP address of the client. NFS versions 3 and 4 support TCP instead of UDP as the transport mechanism, and you can include "tcp" in the options when you export or share file systems, forcing the use of TCP.

Host-based access control does not solve the problem of authentication. For that, real authentication, based on cryptography, is required.

Authentic

Even before Van Doorn's nfsshell appeared, people were concerned about the lack of authentication in NFS. Sun Microsystems developed AUTH_DH, where DH stands for Diffie-Hellman key exchange. (AUTH_DH used to be called AUTH_DES, but I am using its more modern label.) Instead of simply relying on a UID, AUTH_DH uses Diffie-Hellman to exchange session keys for each user. The session key is used to encrypt a pair of timestamps that are included in the RCP header. If the decryption fails, or the timestamp falls outside a five-minute window, then the request is considered unauthenticated.

AUTH_DH relies either on NIS to distribute public and private keys or on the manual distribution of the

/etc/publickey file to all NFS servers and clients. Every time a user changes her password, the public key file must be updated, so practical use of AUTH_DH seems to imply use of NIS as well. You can read more about using AUTH_DH in O'Reilly's *Managing NFS and NIS*.

Generic Security Services for RPC, or RPCSEC_GSS, represents the emerging alternative to AUTH_DH. GSS can support not only authentication but also integrity and privacy, and support for all three have been added to NFSv4 and in some cases are included in patches for older versions of NFS. GSS can use multiple security providers, with Kerberos v5 being the most common.

Setting up a Kerberos Domain is in many ways akin to setting up NIS servers, in that you need secure systems to run the Kerberos Distribution Center (KDC), as well as slave servers for backups. I had hoped to torture myself by setting up a KDC for this column but, fortunately, ran out of time. If you have a Kerberos infrastructure or have been moved by a real desire for NFS security to set one up, by adding krb5 to the export or share options, you can inform your NFS servers to permit access only to NFS clients that can handle Kerberos v5 authentication.

NFSv4 is not limited to Kerberos support for authentication; it is also beginning to include support for SSL-style authentication using LIPKEY/SPKM plugged into the lower layers of GSS instead of Kerberos. The advantage to SPKM is that NFS RPC requests can be authenticated by installing public key certificates for NFS servers or the CAs for those servers on client systems, similar to the way in which Web browsers include certificates for signing authorities. Using SSL, a session key gets generated and is used to sign RPC headers. Once again, you can add real authentication, but this time without setting up either NIS or Kerberos.

The downside to RPCSEC_GSS is that support for it is still very thin. Sun Microsystems has both clients and servers, and Kerberos support is included in newer versions of HP/UX and with AIX 5.2 (with a requested addition). But support in the Linux and BSD world is not quite there yet. You can visit http://www.linux-nfs.org and volunteer your efforts (or other forms of assistance) or check out CITI for FreeBSD support (http://www.citi.umich.edu/u/rees/). You can also read more about the state of NFS security in Michael Eisler's presentation about it: http://www.nfsconf.com/pres03/eisler.pdf.

There are other things you should do when using NFS. When exporting directories, it is wise to disable the use of set-user-ID and device files on the exporting server. The options nosuid and nodev in the export or share statement handle this and prevent a miscreant who has gotten root on a server from creating an SUID shell that will work on all systems that have mounted the file system. Note that clients can use the same flags when mounting remote file systems, disabling the use of SUID and device files from a possibly compromised remote system.

You might think that perhaps CIFS, Microsoft's Common Internet File System, would be a more secure choice. While it is true that CIFS (supported as Samba on UNIX systems) does include real authentication, it is not the same as NFS. For one thing, CIFS is oriented toward added file shares for one user at a time. That is, instead of mounting a file system for use by all users, you instead mount a file share using a username and password for one user. CIFS does not send plaintext passwords across the network (unless misconfigured), but the challenge-response pairs it does use can be cracked and passwords guessed.

Microsoft responded to this threat by beginning to support (you guessed it) Kerberos v5 with Windows 2000. Note that MS Kerberos v5 has some proprietary extensions added to each ticket that are only of use (or interest to) Microsoft systems. While MS Kerberos could be used to support UNIX systems (as the additional cruft will be ignored), the reverse is not true. Do you really expect some self-respecting UNIX system to understand the wacky, variable-length data structures that Microsoft uses as Security Identifiers (SIDs)? Perhaps some day.

So, sure, you can use CIFS instead of NFS. But really, NFS with Kerberos support is just as secure, and more, uh, UNIXy as well.

I wish I could say more, especially after having managed to disturb at least some of my readers, but that is the state of the art today.

TINA DARMOHRAY

firewalls and fairy tales

Tina Darmohray, contributing editor of *;login:*, currently serves as Stanford's Information Security Officer. She is the editor of the Short Topics booklet *Job Descriptions for System Administrators* and was a founding member of SAGE.

tmd@iwi.com

THE RECENT ISSUE OF A POPULAR security rag had the following ad for a security appliance inside the front cover: "If You Believe Routers Can Secure Your Network, You'll Need Another Fairy Tale to Come True." This kind of sales and marketing FUD has been used to hype the differences in security products for years. With firewalls, in particular, there's some historical background to all the mud slinging which can give consumers some market insight and savvy when shopping for a solution.

Recall that the Internet was the outcome of a cooperative research project. In its early days, the challenge was to get and keep the computers talking to each other. Many professional friendships were made between humans at either end of the "next hop," as manual intervention and tweaking of routing tables kept the Net humming. A spirit of cooperation permeated the network, and guest accounts and remote access were de rigueur.

In 1988 the Morris worm hit the Internet hard. It marked the end of blind trust on the Net. Before the Morris worm, professional gatherings focused on simplifying administration and improving reliability of the machines and the network that connected them. After the worm, attention turned to securing the network as well. Network administrators started cobbling together network access controls. Increasingly, Net News discussions, BoFs, and conferences dealt with the new focus on computer and network security. It became a necessary evil.

Today's booming computer security market masks the early struggles of security-minded individuals. The early implementers of network access control often found themselves not just explaining the technology but defending it as well, as they were perceived as being counter to the spirit of cooperation that had been the norm. Giving an invited talk on network security was sometimes like volunteering to be the person in the dunking booth at the local carnival!

Despite their unpopularity in the user community, firewalls were deployed one network connection at a time. The earliest firewalls consisted of routers configured to filter out packets destined for particular internal network ports, thereby denying access to internal services. To those managing the network connections, this was an obvious countermeasure. The router hardware and software already existed and was in place. All that had to be done was to use an existing capability to implement a more secure stance.

Additionally, since the changes were made at the network layer (layer 3 of the textbook 7-layer network model), they were transparent to users' applications. A packet-filtering router yields a lot of bang for the buck. Even simple access control lists (ACLs) can dramatically enhance site security. And, as with all firewalls, the "conservation of energy" is huge as you protect many internal machines with a small set of perimeter devices.

A simple filtering router examines network packets for source and destination IP address, source and destination port numbers, and protocol type. Using this information, decisions are made to route or reject packets. Administrators choose the ports to be forwarded or dropped based on the way that ports are assigned to applications. A UNIX kernel reserves ports < 1024 for privileged processes; it randomly assigns ports 1024 and above, as needed, to processes that request them. The reserved ports are conventionally assigned as "well-known ports" with a particular service being associated with a particular port number. Thus, rejecting packets destined for a particular well-known port translates into filtering out the corresponding service. For example, rejecting all traffic to TCP port 23 is intended to disallow Telnet connections. Similarly, rejecting traffic for TCP port 80 would disallow HTTP connections, and so on. (For a list of well-known ports and their assignments, see http://www.iana.org/assignments/port-numbers.)

Simple packet filters provided a cheap and easy security solution, but the debate had just begun. Many argued that they didn't provide enough protection. Like most security solutions, firewalls are about narrowing the window of opportunity for intrusion. Critics pointed out that packet-filtering firewalls innately fell short of application-level firewalls. For the most part, the criticisms focused on the tenuous binding of port numbers to applications. Recall that this binding is based on the UNIX convention of assigning ports. This is not a standard that must be rigidly adhered to, but a convention, a traditional way of doing things. To that end, there's nothing stopping someone from running a service on a different-than-expected port and thereby slipping around the packet filter. The most common example of this is the abuse of the high-end ports, numbered 1024 and higher. These ports must be allowed through static filters in order to accept the return traffic of internally initiated connections.

My favorite story on this subject concerns a battered university administrator who implemented a few simple packet filters to disallow some of the most dangerous services, like inbound Telnet. Predictably, the students rebelled and publicized their own version of the well-known port assignment convention, with Telnet found on the port number matching the last four digits of someone's telephone extension. Most of the new port assignments were > 1024, and glided right through the packet filter, much to the dismay of the security-minded network administrator.

The critics of packet-filtering firewalls usually fell in the application proxy firewall camp. They argued that proxy firewalls are more secure because, since they work at the application layer, they could examine the payload of the packet, enabling them to be protocol-specific. They could also authenticate traffic at the user level. However, in those early firewall days, purists who felt that packet filters just weren't secure enough were forced to build their own proxy firewalls, since there were none commercially available at that time.

Writing proxy firewalls isn't for everyone. There's real coding involved and, since it's for security purposes, it better be well written and torture-tested. That's too tall an order for many system administrators, so packet filters remained a popular alternative. However, some able and willing administrators took the time to write proxies. A few of those early efforts were the DEC SEAL, SOCKS, and the TIS Firewall Toolkit. The latter two played a key role in what happened next in the evolution of firewalls.

When SGI acquired MIPS one afternoon in 1992, the MIPS proxy firewall named SOCKS[1] became publicly available. Finally, a proxy firewall was available to the masses. SOCKS was written by David Koblas, a system administrator for MIPS Computer Systems. It is an elegant solution, ultimately providing a mechanism to protect internal systems by means of transparent proxies that operate at the TCP protocol level.

However, there was a hitch. The transparency came at a price: Software needed to be deployed on each client computer. While this was reasonable for the small homogeneous MIPS Computer network, it can be logistically prohibitive on any large-scale network with multiple OS platforms.

In 1994, the White House was shopping for a vendor who could get it onto the Internet securely. Marcus Ranum, then of Trusted Information Systems (TIS), was up to the task. He brought up whitehouse.gov and in the process developed the TIS Firewall Toolkit (FWTK). The FWTK proxies actually implemented a secure subset of the most common network application protocols. Because his work was funded with taxpayer money, Marcus decided to release a version of the FWTK to the Internet community. The release wound up being an ingenious marketing move that gained a huge following for the TIS technology. It didn't require any software modifications on client systems and therefore ran quite easily in heterogeneous environments. But the lack of software mods came at the price of transparency to the users. Users were exposed to the "double hop" it took to get out of the

network via the proxy firewall. For example, to Telnet outbound from the protected network, a user would Telnet to the proxy server and then issue the command to connect to the final destination.

While functional, this wasn't nearly as elegant as the entirely behind-the-scenes connections that transparent proxies like SOCKS made on the users' behalf. A further drawback to the true application proxy approach was the necessity of providing a specially coded proxy program for each protocol that was to traverse the firewall. With the rapid proliferation of protocols in the latter half of the 1990s, proxy firewall vendors were forced to play a continuous game of catch-up. So proxy firewall fans were left to port code or provoke users, with no real alternatives in between.

Meanwhile, vendors began to warm up to the emerging firewall market. Livingston Enterprises was among the first to market routers as firewalls. They heeded Brent Chapman's call for more security functionality and implemented the features he outlined as being critical (and missing in most vendors' routers).[2]

At the time, and for a good while after Livingston's early entry into the firewall market, leading vendors such as Cisco didn't provide the increased functionality that industry spokesmen like Brent were calling for. Cisco eventually caught up, but their slow market entry was a black eye for them with the security enthusiasts for years, and a soft underbelly on which proxy firewall proponents hammered.

Probably the most innovative of the early vendor entries in the market was Checkpoint's stateful packet-filtering firewall. Checkpoint introduced a packet-filtering product that considered connection-state information when deciding to pass or drop traffic. Tracking connection state (including "virtual" connections for nominally connectionless protocols like UDP) enables you to permit traffic to pass through the packet-filtering firewall only if it is associated with a connection you've explicitly approved, such as the data connection of an FTP session, or the UDP response to a DNS query. Checkpoint also introduced dynamic packet filtering, which opens only the ports you need at the time you need them. So, for example, if you need to permit traffic on a high-level port in response to an outbound connection request, that port is allowed, but only for the duration of the connection. These two major improvements to vanilla packet filtering really raised the security bar for packet-filtering solutions.

Aside from these early breakthroughs, there hasn't been a huge change in the bottom line of firewalls in more than a decade: packet-filtering firewalls look at layer 3 information, while proxies are able to inspect the actual content of the packets. Once you open the payload portion of the packet, you can make decisions on that content as well, e.g., on user identity or whether the data matches the kind of bits that should be associated with a specific protocol. Beyond that, though, most of the "new" entries into this market space are just new marketing variations on the old theme. Eventually the commercial version of the Firewall Toolkit, Gauntlet, developed transparent proxies and incorporated packet-filtering features to respond to its perceived failings. Packet-filter vendors added the ability to filter on packet payload. And everyone moved toward "appliance" firewalls and Web GUIs for administration.

TIS's Gauntlet and the Livingston Firewall router are gone, and a horde of new names have taken their place. The mainstream commercial firewall products available today are actually hybrids of the packet filters and application proxies first developed in the early 1990s. "Deep Inspection" firewalls are just packet filters that are going a little further up the stack than traditional packet filters. "Intrusion Prevention" devices are usually combinations of signature-based intrusion detection systems and traditional firewall technology that shut down connections based on patterns in their payload, in addition to making decisions based upon ports or protocols.

What is true is that solutions continue to merge, as each of the two primary areas takes a little more from one or the other and incorporates it into its baseline. A close look at the new marketing rarely reveals actual new technology, though.

Next time you see a new firewall advertisement, take a step back and analyze the claims through the lens of history before you decide to fear or to fantasize that fairy tales have come true.

1. D. Koblas, "SOCKS," *Proceedings of the Third USENIX UNIX Security Symposium* (Baltimore, MD: USENIX Association, September 1992).

2. D. Brent Chapman, "Network (In)Security Through IP Packet Filtering," *Proceedings of the Third USENIX UNIX Security Symposium* (Baltimore, MD: USENIX Association, September 1992).

STEVEN ALEXANDER

the importance of securing workstations

Steven is a programmer for Merced College. He manages the college's intrusion detection system.

alexander.s@mccd.edu

SECURING WORKSTATIONS IS AS important as securing servers. Even so, the security of workstations is often ignored, because servers are individually more important.

Eyes on the Prize

Most of the attention given to computer security by system and network administrators focuses on servers and network devices. Allocating the bulk of a network administrator's resources to these systems makes sense, since a failure or security breach in one of them has the furthest-reaching consequences. Some recent privacy laws (such as SB 1386 in California) require the disclosure of security breaches where personal information may have been disclosed. Such laws are likely to reinforce the focus on server security. It is important, however, to implement and maintain comparable security measures for workstations.

Much of the attention given to PC security has focused on viruses, worms, and spyware, since such malware can affect productivity. Unfortunately, too many organizations fail to consider the other threats to PC systems and the consequences of a successful security breach by a person, rather than by a random automated attack.

Some of the most lucrative targets for data thieves are large database servers, but workstations also contain valuable information (including passwords for servers). It is important to note that not all workstations (or servers) are created equal. It may only be necessary to implement minimal security measures in a computer lab at a college or university—provided, of course, that the lab machines are kept separate from the rest of the network. The machines in a computer lab have little information that is of value to an attacker. Of course, an attacker can still use the machines to launch other attacks, trade pirated software, or snoop on a student checking his email. On the other hand, a workstation belonging to someone in the human resources or payroll departments might be a worthwhile prize by itself. Such systems often contain an abundant amount of valuable information which, if compromised, might lead to an embarrassing public disclosure with financial consequences.

Employees in different areas of an organization might have many sorts of valuable information on their workstations, including payroll and tax information, company financial data, strategic and planning information, confidential memos, and more. To make matters worse, access to this data is harder—if not impos-

sible—to audit than is access to a centralized database server. Encouraging employees to store as little as possible on their own workstations might have a small positive effect, but such efforts can be obstructed by real-world needs, as well as by the fact that an attacker is likely to gain access to the same centralized resources as the owner of a compromised machine.

IT departments can also pose a major risk. Not only do programmers and other IT employees often have valuable data on their machines, their other duties often introduce extra security risks. It is not unusual for IT staff to run a wide variety of applications, test out new third-party software of various sorts, and test internally produced software. This software, particularly if it is a networked application, can be a security risk. In larger IT departments, the job responsibilities of individual employees might be well separated (though not necessarily), but in small departments employees are often required to take on responsibilities that might properly belong to several different positions. Part of this problem can be alleviated by using separate machines to test software, but too often this is not an option.

The security of an individual workstation is unlikely to be as important as the security of many of the servers in an organization. The security of all workstations together, however, might be as important as the security of all servers. While an individual server can hold more data, breaking into a workstation is often easier and may provide a larger reward for the effort expended. It may also be used as a foothold to a larger system.

Building Secure Systems

It is essential to secure new machines and to put management and patching procedures into place before giving the machines to employees or connecting them to the network. The folks at the San Diego Supercomputing Center have done some admirable work in this area. Abe Singer's "Life Without Firewalls" discusses this work and is required reading.[1]

All new systems should be fully patched, and automated patching should be set up. Windows Automatic Update can be useful, but many administrators (particularly in larger organizations) would do well to use Microsoft's SUS (Software Update Services) or SMS (Systems Management Server). Antivirus software should be installed on all Windows machines; it is less of a concern for other platforms. Anti-spyware tools should be used as well.

Additional security measures must also be put into place on each workstation. Unnecessary services should be turned off, default accounts should be disabled or the passwords changed, etc. Most of the "best practices" applied to servers apply to workstations. The Center for Internet Security has published useful guides for Linux, FreeBSD, Solaris, Windows, and other systems.[2]

Open source systems should use buffer overflow protection. Many books and articles about security say little, if anything, about this. They should! The amount of protection used depends on the requirements of the system and on considerations such as performance. Compiler patches such as StackGuard and ProPolice/SSP provide good protection with a minor performance impact. The addition of OS-level protection such as PaX and W^X provides much better security but at a higher performance cost. The SmashGuard Web site has information about several buffer overflow protection mechanisms.[3] Information about W^X is available from the OpenBSD site.[4]

The NSA has produced guidelines for Windows 2000 and XP.[5] The recommendations are somewhat restrictive, so most administrators would do best to study the NSA and CIS suggestions and then develop their own policy. Pay attention to network logon rights, terminal services access, and remote registry access

even if host firewalls are used. Windows administrators should also study the new features available in Windows XP Service Pack 2. The Windows XP firewall is not very flexible but is adequate for many networks and, if manageable, it should be used. Once a reference system has been constructed, new machines can be built using tools such as Ghost or Altiris to clone them. Reference configurations can be maintained on UNIX using cfengine.

In most circumstances, users should not be given full administrative rights to their own machines. The more privileges a user has, the more likely that person is to use those privileges to circumvent security measures. What an administrator sees as necessary the user may see as irksome. If users must be given local administrative rights (more common in academic than business environments), measures can be taken to restrict certain prohibited software (such as Kazaa or edonkey). There are a number of ways this can be accomplished; for instance, software can be restricted directly using MS Group Policy and indirectly by preventing network traffic using firewalls. Tools such as Altiris can be used to inventory the software installed on workstations throughout the organization.

Data loss is important to consider. It can come about through an intelligent attack or by simple hardware failure. Network storage can be used to mitigate this problem. Each user should have his or her own folder on the network, which can be used as a repository for important documents and data. The user folders should be backed up regularly.

Network Security

Firewalls have taken a beating at the hands of several security experts in the past few years. One of the major reasons is that many people (technical folks as well as managers) think that firewalls are a cure-all; instead, because of the way they are used, they become a palliative. Many people think it is okay to put a firewall on the border of a network, ignore everything on the inside, pat themselves on the back, and announce, "We're secure. We have a firewall." Shame on them! May they find themselves in the company of a BOFH and an empty tape safe.

Firewalls do not (and never will) block out all of the bad traffic while allowing well-intentioned, legitimate users to access the network. Firewalls can be used to restrict the types of traffic that are allowed through, though, thus narrowing the window of vulnerability. By enforcing certain restrictions, firewalls require attackers to have a greater degree of skill or luck in order to launch a successful attack. Often, as is the case for much of what I discuss here, the firewall is a router with packet-filtering capabilities.

Firewalls should be used at the border of a network to prevent or hinder reconnaissance and to prevent access to services and machines that should not be accessed from outside the network (such as internal DNS and FTP servers). Most (but not all) ICMP traffic should be blocked, thus preventing a lot of reconnaissance activities. Unfortunately, blocking all ICMP breaks things. For instance, many administrators have caused problems by blocking the "fragmentation needed" ICMP packets that are required by Path MTU Discovery.[6] Certain services that must be accessible to the outside should be placed on a screened subnet so that a compromise of one of them poses less of a threat to the rest of the network.

Critical divisions within the network should be separated from each other. Departments should be logically separated and traffic between them controlled. Whenever possible, a user should be unable to use his or her workstation to access workstations of users in other departments. If collaboration is required between departments, shared storage should be set up on a server that members of both departments can access.

Separation can be achieved in a number of ways. VLANs can be used but have a number of issues.[7] Firewalls are more flexible but can be difficult to configure correctly. The problem with firewalls is that IP addresses are unauthenticated. Just because an incoming IP address matches the one used by Debbie Sipiyae in Accounting doesn't mean that the packet wasn't generated by Joe Student in the computer lab. Vulnerability to IP spoofing can be mitigated by using ingress and egress filtering. This filtering should be used at the network border and between segments within the network. This won't prevent all address spoofing, of course; a user in Accounting will still be able to spoof the address of someone else in Accounting, but he shouldn't be able to use an IP address that belongs to Marketing, HR, or IT.

System administrators may need to access workstations throughout the organization. If possible, this access should be restricted to certain administrative workstations and servers rather than allowing all IT personnel to have this network access. Because administrators require such open network access, system administrators (and possibly support staff) should be placed on a different network segment from other IT staff (such as programmers), who do not need unfettered access to the rest of the network.

When a workstation is compromised, the accounts of the users who use that workstation are usually compromised as well. Furthermore, access to one system on a network is often used to gain access to others systems and accounts. These risks are greatly reduced by not using plaintext passwords and by using solid password encryption. Abe Singer wrote a *;login:* article about eliminating plaintext passwords,[8] and I talked about password encryption in another *;login:* article.[9] Many system administrators still seem to believe that sniffing is difficult or impossible on switched (as opposed to hub) networks, but this is not so.[10]

Conclusion

The importance of information does not vary according to the machine the information resides on. A file containing names and social security numbers is just as valuable whether it is stored on a highly secure file server or a Windows PC. Owing to the fact that administrators do not know in advance what information will be used or stored by each of the users of an organization, the security of each user's machine should be as strong as possible.

REFERENCES

1. Abe Singer, "Life Without Firewalls," *;login:*, vol. 28, no. 6, December 2003, pp. 34–41. See also Singer's "Tempting Fate" in this issue.

2. Center for Internet Security, *http://www.cisecurity.org/*.

3. SmashGuard, *http://www.smashguard.org/*.

4. OpenBSD, *http://www.openbsd.org/papers/*.

5. NSA Information Assurance, *http://www.nsa.gov/ia/*.

6. Richard van den Berg and Phil Dibowitz, "Over-Zealous Security Administrators Are Breaking the Internet," *Proceedings of the 16th Systems Administration Conference (LISA '02)*, November 2002, *http://www.usenix.org/publications/library/proceedings/lisa02/tech/vanderberg.html*.

7. Rik Farrow, "Network Defense: VLAN Insecurity," March 2003, *http://www.spirit.com/Network/net0103.html*.

8. Abe Singer, "No Plaintext Passwords," *;login:*, vol. 26, no. 7 (November 2001), pp. 83–91.

9. Steven Alexander, "Password Protection for Modern Operating Systems," *;login:*, vol. 29, no. 3 (June 2004), pp. 23–33.

10. Dug Song, "dsniff," *http://monkey.org/~dugsong/dsniff/*.

ABE SINGER

tempting fate

Abe Singer has been a computer security researcher with the Security Technologies Group at the San Diego Supercomputer Center for the past five years. His work has involved growing SDSC logging infrastructure and analysis capabilities, participating in incident response and investigation, and working with the TeraGrid Security Working Group. Mr. Singer, with Tina Bird, is the author of *Building a Logging Infrastructure,* SAGE Short Topics booklet #12. Mr. Singer's current research is in automation of syslog parsing and analysis toward data mining of logs for security. In addition to his work at SDSC, Mr. Singer is an occasional consultant, expert witness, and lecturer. Prior to SDSC, he was a consultant for several years and a programmer and system administrator for over 15 years.

abe@sdsc.edu

IN THE DECEMBER 2003 ISSUE OF *;login:* I wrote an article called "Life Without Firewalls" in which I talked about how we do security at SDSC, why we do not use firewalls, and how we have been very successful at keeping out intruders.

If I were superstitious, I'd say I should have known better than to tempt fate. In the Spring of 2004, SDSC had an intrusion that gave us a pretty good amount of grief.

Rik Farrow suggested that I call this article "Eating Crow," but I stand by what I said in my previous article. The intrusion was successful only because we didn't follow our own rules well enough. Our strategy helped us detect the intruder quickly and reduced the scope of the intrusion (it could have been much worse, and for some other sites it was). Eight of our hosts (out of several hundred) had root compromises; moreover, the intruder was able to modify user-owned files on one of our NFS servers. Our reference system model allowed us to have the compromised hosts reinstalled and up and running in less than two hours each—we didn't have to think twice about reinstalling a host.

As for our lack of firewalls, in this case a firewall would not really have helped (as I'll explain below). In fact, shortly after the attacks, Marcus Ranum sent me an email saying simply, "Living without Firewalls . . . ;-)," so I explained what happened, to which he responded, "aw crap, transitive trust, gets 'em every time."

So I'll explain how our intruder got it, where we failed, and how our security strategy helped mitigate the problem; I'll also talk about what we've learned and what we're doing differently. I'm going to be deliberately vague about some things, to protect the privacy of some of the people who have been compromised and because the intruder is still actively attacking sites.

Beginning in December 2004, we started hearing about compromises at other sites. The intruder had gotten root on some machines and had installed a trojaned SSH client, which he used to gather usernames and passwords to other sites as users logged on to the compromised hosts and then SSHed into remote sites.[1]

The intruder would then log in to a user's account at the remote site and look around for ways to compromise the host or any other host at the remote site.

By March 2004, we knew of several sites that had compromises. Tina Bird published a bulletin which described the activity at Stanford and elsewhere.[2]

One morning in late March I received an email from one of our systems about a failed sudo attempt by one of our sysadmins (our version of sudo sends email to root when it fails). A quick phone call verified that it indeed was not the sysadmin. The host on which this happened was the host we use to manage the rest of our machines—those machines allow root rsh from the management host so that we can automate configuration of multiple hosts (this was described in my previous article). Of course, this definitely got our attention.

A quick check of logs found an rlogin[3] to that account from a workstation. A ps on that workstation showed a root-owned process called "foosh"—definitely a bad sign.

The process disappeared within minutes of our looking at the host. We believe the intruder had spotted us and decided to leave.

A check of the logs showed a particular user logging in (via SSH) to one of our workstations from a remote site with which we collaborate, and within a minute that user logged in again from a cable modem somewhere in the Pacific Northwest. Following immediately were logins from the workstation to every other system on which the user had an account (and numerous failed attempts on hosts where the user did not have an account).

Our logs also showed, shortly after the rash of logins, that root was su'ing to several users, including the user who had initially gotten my attention. The tty from which the su's were executed corresponded to one that the suspected user had logged in to, and that user didn't have any privileged access. This was our indication that the intruder had definitely gotten root on a host.

Various log entries gave us a clue about what the attacker was doing. He had managed to get root on a system and had done some investigating to determine who might have privileged access. We think he looked at things like who was in the root group, the SSH known_hosts file, etc. He targeted our management system, which only has accounts for those users who need it. Thus, he needed an account on that host for which he didn't already have the password. He picked a user who had access and su'ed to that user in order to be able to rlogin to the management system. Once there, the intruder apparently tried sudo, hoping to take advantage of cached sudo credentials, which failed.

So how did the intruder get root on the first host? We're pretty sure he used a local kernel exploit. We had patched the host but had not rebooted it, so the patch had not actually taken effect (Mistake #1). We figured this not only from our knowledge of the patch state of the host, but because the intruder placed an executable in a user's home directory and modified the user's .cshrc to try to run the binary anytime that user logged in to a Solaris box. The intruder was trying to get users to root boxes for him. Fortunately, the intruder wasn't very good at writing shell scripts, as there was a syntax error in the .cshrc file.

The logs also showed the intruder trying things like putting a "+" in .rhosts (which is disabled on our hosts and is logged when an rlogin is attempted). We found keys added to various users' .ssh/authorized_hosts file. Process accounting also showed the intruder running a program called "n," but he had erased his tools before logging out, so at the time we didn't know what it was.

We rebuilt the machine and rebooted others that had the same patch applied. We checked all users' authorized_hosts and .rhosts files to make sure there were no other accounts accessible. We changed the password for the known compromised account.

I also called the site from which the compromised user had originally logged in—where the password had been intercepted—to let them know they probably had a compromise. They called me back a couple of hours later and confirmed that they had been owned.

A couple of days later, we found some modified authorized_keys files again and discovered that another Solaris host was compromised. This host was not vulnerable to the same exploit that had been previously used, so the intruder had another exploit. This host had also been patched but not rebooted. We did a lather, rinse, repeat—reinstalled and rebooted hosts that needed it. This host was also a Solaris 8 host, as was the first host compromised, so we decided that all Solaris 8 hosts needed to be patched and rebooted (Mistake #2).

That weekend we discovered the intruder had gotten root on a Solaris 9 box. We then realized we needed to make sure all of our hosts were fully patched and rebooted. Machines were carefully rebooted one by one to make sure they came up okay, and at midnight a few hosts that ran special applications were left for the sysadmins who administered them to reboot (Mistake #3).

The next morning, before the machines had been rebooted, the intruder got root on another host, su'ed to yet another user, and sent an email out to every email address at SDSC and UCSD that he could find, with some rather rude ASCII art and some typical script-kiddie language, talking about how great he was and what losers we were.

That was embarrassing, but we were able to recover, reinstall the host that was compromised, reboot everything, and make sure all our patches were up to snuff.

After that, we did not discover any more root compromises on our Solaris or Linux hosts (I'll qualify that with "that we could detect"). News of the various intrusions spread, and we started getting calls from the press. A spokesperson was appointed to deal with the press. He was quoted as saying that the attacker had only gotten a few perimeter systems and had not gotten at our infrastructure (Mistake #4), which was true at the time: A few workstations had been compromised, but our file servers were intact, and we had no indication that the intruder had gotten onto any of those hosts (although we did have some indications that he tried).

The day after the newspaper article came out, the intruder got root on the login node of one of our supercomputers, did a "wall" to all the users with some more ASCII art, and did a "shutdown" on the host. In the "wall" message, he quoted the part of the news story about not getting at our infrastructure and claimed that this shutdown was proof that he *had* actually gotten at our infrastructure. We believe the intruder exploited an unpatched FTP server that shouldn't have been running on the host in the first place (Mistake #5).

So that system was taken offline. It was due to go out of production in a few weeks anyways, so we just left it offline.

Somewhere in the mix of this, I received a call from someone at another university. They had found John-the-Ripper running on a cluster of theirs, with what appeared to be a fragment of our shadow password file, including my encrypted password. He sent me a copy, and I was able to confirm that it was indeed my password.

Thus, we also knew that the intruder was cracking passwords in addition to running trojaned SSH clients at other sites.

When we took down the supercomputer, we also changed passwords for all users (several thousand) and audited our other supercomputers to make sure that there weren't signs of a compromise.

From then on, things calmed down. We continued to see the intruder log in to compromised accounts (even with the password changes), but no sign of root compromise. This was an *acceptable* state, not a great state, but we could live with it; our big concern was root compromise and compromise of our infrastructure—the file servers, DNS servers, and such.

I then received a call from someone at yet another university. He had found what appeared to be a copy of my email inbox. He gave me the header timestamps from the first and last message, and they corresponded to messages in my inbox. And the dates were from a couple of weeks *after* we had cleaned everything up. We had no idea at the time how the intruder had accessed it—there were no signs of root compromise anywhere. And we didn't know how my account could have been compromised, as I had not logged in from other sites—had not left credentials available for use—and the logs did not show any suspicious logins to my account. A log message showed some more failed attempts to use "+" in my .rhosts file, but we didn't know how it had been put there. The assumption was that my password had somehow been compromised. So I changed all my credentials and started logging in only from my laptop using an SSH private key that had never left the laptop. (It's a Mac OS9 laptop, so I felt pretty good about the integrity of the OS.)

Finally, a few weeks later, we figured out how the intruder had done it. We had heard of a well-known tool called nfsshell4 that the intruder had used at some other sites, and so we tried it at our site. It worked. The nfsshell tool exploits NFS servers that allow clients to send mount requests from an unprivileged ("high") source port. And once a file system is mounted, there is no validation of the UID used in NFS file operations. In other words, nfsshell allows an unprivileged user to mount a file system and then read and write files using any UID. Since we squash root access via NFS, the intruder was unable to write files as root, but was able to write to files owned by any other user. So that's how he was able to stick SSH keys into authorized_keys files, and how he got at my inbox, etc.

While we were looking at this problem, but before we could react with a fix, the intruder decided to erase a couple of home directories. At that point we took SDSC off the Internet until we could remedy the situation.

It turns out there was a simple kernel parameter that had to be set to disable unprivileged ports, and that parameter had not been set. nfsshell is a very old exploit, and our previous server had been immune; we had made the assumption that the new one was equally immune (Mistake #6).

We fixed the file server, cleaned up, and brought SDSC back online.

Since then, we continue to have the occasional user account compromise, but no new root compromises. We have seen the attacker come back and try the same exploits, with no success.

Based on our lessons learned, we have changed a few things at SDSC: We have shortened our patch cycle (it was about a month long), and reboots get priority over uptime. We have implemented two-factor (token-based) authentication on our critical infrastructure machines, as the only people who have to access those are system administrators.[5]

So now you're probably wondering why I stand by my previous article and why I say a firewall would not have helped. If we had had a firewall, we would have had to

allow inbound SSH, so that our users could log in. The intruder used the same access mechanism our users do, often from the same outside hosts.

Second, as I indicated at the beginning of this article, we mostly were owned because of things we *should* have been doing. It does show, however, that doing it right is *hard*. You have to, well, do *all* of it right. If we'd had our machines fully patched and rebooted, the intruder probably wouldn't have gotten root, at least not with the techniques that we saw him use.

Also, many of the things we do mitigated the extent of the compromise. For instance, squashing root and set on our file servers kept the intruder from creating setuid-root files and running them on other hosts.

Having Kerberized sudo kept the intruder from getting sudo privileges, even when he managed to obtain a user's password. Having a separate set of credentials for privileged access is definitely a Good Thing.

Our centralized log server allowed us to look at log activity across hosts and provided assurance that we had a good copy of log information, even if the intruder erased or altered logs on the compromised host.

And having reference systems allowed us to recover quickly. Our Solaris hosts take a couple of hours to reinstall, mostly unattended (it takes a while for a new host to install all of its patches). Our Linux hosts take around half an hour.

We fared better than some other sites. For instance, we know of one site that allowed root logins via SSH for sysadmin purposes. The intruder owned the workstations that the intruders were using to manage other systems, and from there just got the root password. Another site that we know of did not have root squashed on their NFS server, so the intruder was able to create setuid-root programs from one host and then log on to another host and run the program. Yet another site had to rebuild all of their compromised hosts by hand—they had not automated the process.

Some lessons we learned about intrusions: Never underestimate the capabilities of your attacker. Assume your communications are being monitored. Don't taunt the animals. And reboot everything.

REFERENCES

1. In the academic community this is a very common practice; organizations are involved in collaborative activities, researchers may be at one institution and be using the facilities at another, etc. In fact, most of SDSC's several thousand users are located at other institutions, as our business is to provide computing resources to researchers.

2. Stanford University Information Technology Systems and Services, "Security Bulletin on Multiple UNIX Compromises," http://securecomputing.stanford.edu/alerts/multiple-unix-6apr2004.html.

3. I often get funny looks when I mention using rlogin. We *only* allow r-commands between hosts that we manage, and we only allow managed hosts on the networks that this traffic travels over. So spoofing tcp connections requires access to the local network.

4. nfsshell, ftp://ftp.cs.vu.nl/pub/leendert/nfsshell.tar.gz.

5. I still believe that token-based authentication is too much of an expense (not just the hardware costs, but the support overhead) to do for all users, but it's appropriate for those who have privileges.

ROBERT HASKINS

ISPadmin

Robert Haskins has been a UNIX system administrator since graduating from the University of Maine with a B.A. in computer science. Robert is employed by Renesys Corporation, a leader in real-time Internet connectivity monitoring and reporting. He is lead author of *Slamming Spam: A Guide for System Administrators*.

rhaskins@usenix.org

THIS ARTICLE IS BASED ON THE NEW book *Slamming Spam: A Guide for System Administrators* (ISBN 0-13-146716-6) by Robert Haskins and Dale Nielsen. This material is copyright 2005 Addison-Wesley Professional, all rights reserved. It is reprinted with permission of the publisher, Addison-Wesley Professional. This material is taken from Chapter 12 and is identical to the Camram section in the book, except that Figures 4 through 8 have been deleted for space reasons.

Camram is a "sender verification" system, similar to challenge/response systems TMDA and ASK. It has a very nice Web-based interface to CRM114 in addition to its native sender verification functionality. The idea is any message that is not from a sender who computes a certain algorithm (using a Hashcash) is processed through CRM114. Any message that doesn't have the computation result in the headers must be analyzed by CRM114.

While sender verification is controversial within the anti-spam community, these types of systems are useful to some people. Camram might be used in any installation that desired a graphical, Web-based interface to CRM114. It also could be used at a site where additional protection beyond traditional header/content analysis (such as SpamAssassin or bogofilter) was desired. If enough email originators use sender compute headers, impact on recipient Camram email infrastructure would be reduced, due to the fact that those messages with sender compute headers bypass the more resource-intensive CRM114 checks.

For more information on Camram, see *http://www.camram.org*.

Camram

The reason for Camram's original implementation was as a reference implementation for a sender compute system, namely Hashcash. Although this is still a large part of the goal, Camram has tight integration with the CRM114 spam classifier. It also contains a graphical user interface to manage itself and the CRM114 application as well. Camram is worth implementing just for the ease of use it provides in managing CRM114.

Camram can be set up as an invisible proxy between your existing MTA and email systems that want to send your users email. This eliminates the need to

run Camram on your existing (perhaps overly loaded) email systems. Camram refers to this setup as the interception method. You should be aware that Camram is still a work-in-progress. Some of the functionality doesn't work precisely as expected, but it should be suitable for most situations. Be sure to check the Camram Web site often for code updates.

INBOUND MESSAGES

You can deploy Camram in two different ways in your inbound email infrastructure. The first way is by using procmail to redirect incoming messages, in a setup where Camram is run on the same machine as the end user mailboxes. This is the setup we cover here.

The second method that can be used is interception. This method "intercepts" the SMTP port 25 connection and redirects it to the Camram server, which processes the message and sends it to the mailbox. The interception method is used in a situation where your organization's email system is distributed into machines that perform the email relay function and servers that house mailboxes. Another case is when your primary server is Exchange/Domino, where you cannot run Camram directly on the mail server. Implementing an anti-spam solution such as Camram on a separate system helps to distribute the load on machines outside of your regular mail machines.

In either case, the actual processing of messages is the same, regardless of whether the procmail or interception methods are used. Figure 1 shows the flow of messages through the Camram system.

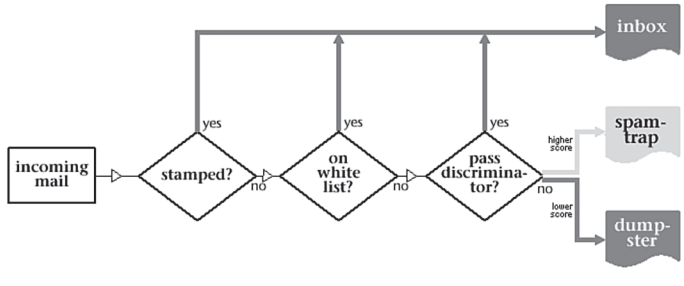


FIGURE 1.

Camram inbound message flow. (From http://www.camram.org; courtesy of Keith Dawson, dawson@worlds.std.com; used with permission.)

OUTBOUND MESSAGES

Messages leaving the Camram system must be stamped to show that they have been processed through the Hashcash computational system (see Figure 2). This is done as a proxy, using the EmailRelay software. The message is reinjected into the MTA on port 30025.

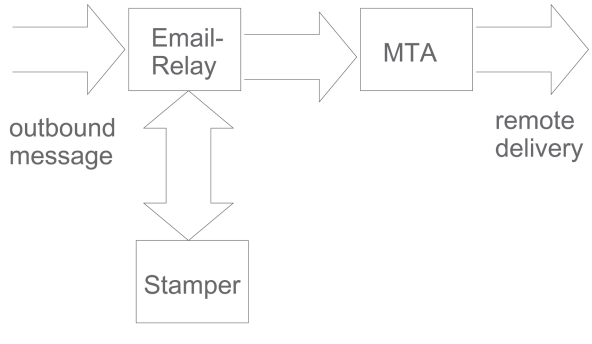


FIGURE 2.

Camram outbound message flow.

Installation

Camram can be downloaded from http://www.camram.org/download.html. We cover Camram version 0.3.25 here. The build script downloads all of the needed components, including:

- TRE—Regular Expression matching library required by CRM114
- CRM114—The Controlled Regular Expression Mutilator covered in Chapter 8, "Bayesian Filtering"
- Hashcash—Implements the sender compute algorithms required by Camram
- EmailRelay—MTA used by Camram to implement its message stamper functionality
- normalizemime—Used by CRM114 to convert MIME-encoded text

These are external packages that Camram requires for operation. Camram will download and install Python if it is not available on the system or if it is not at the correct version level when you run the buildit.sh script (shown next). After downloading, become root, extract the files, add the Camram group and user, and run the build script like this (the downloaded installation is assumed to be /usr/local/src/raging_dormouse-0.3.25.tar.gz):

```
bash$ sudo su
# mkdir /usr/local/src/camram-0.3.25
# cd /usr/local/src/camram-0.3.25
# groupadd camram
# useradd -g camram -m -d /usr/local/camram camram
# tar xzvf ../raging_dormouse-0.3.25.tar.gz
# mv raging_dormouse-0.3.25/* .
# bash buildit.sh
```

You may need to restart the download script if a download error takes place. The raging_dormouse release will exit the build process if there is a checksum error in one of the components. The build script will make sure that the appropriate third-party applications have been downloaded before continuing on.

After the initial setup script has been run, several additional steps need to take place. These actions include:

- Setting up the Camram GUI for use under Apache
- Setting up the MTA (Sendmail) to work with Camram
- Configuring a Procmail recipe for use with Camram

APACHE INSTALLATION

Next, install the Camram hooks for the Apache Web server. The installer attempts to copy the configuration to the Apache configuration directory on some Linux distributions, namely /etc/httpd/conf. If this is not how Apache is set up on your system (for example, Debian), then copy the configuration file manually to the Apache configuration directory and restart Apache like this:

```
# cp -p /usr/local/camram/ancillary/camram.conf/etc/apache/
  camram.conf
# /etc/init.d/apache restart
```

SENDMAIL (MTA) INTEGRATION

Integrating Camram with Sendmail requires setting up Sendmail to listen on three IP addresses and ports: we use 127.0.0.1 port 25, 127.0.0.1 port 30025, and the publicly available inbound interface. Any available IP and port combination can be used, but these are what Camram recommends, so they are the ones we use.

If you set up Sendmail per our examples in other parts of this book, sendmail.mc is located in /usr/local/src/sendmail-8.12.11/cf/cf/. If your current configuration is sendmail.cf, then edit your sendmail.mc file and add the following three lines, replacing 192.168.16.9 with the public IP address of your Camram machine that accepts email from the Internet:

```
DAEMON_OPTIONS('Port=smtp,Addr=192.168.16.9, Name=MTA')dnl
DAEMON_OPTIONS('Port=smtp,Addr=127.0.0.1, Name=MTA')dnl
DAEMON_OPTIONS('Port=30025,Addr=127.0.0.1, Name=MTA')dnl
```

These lines tell Sendmail to listen to port 25 on its public IP address and localhost address (127.0.0.1), as well as 30025 on localhost for reinjecting messages into the MTA. Then rebuild sendmail.cf, install it (saving the old one), and restart Sendmail:

```
bash$ sudo su
# cd /usr/local/src/sendmail-8.12.11/cf/cf/
# make sendmail.cf
# cp /etc/mail/sendmail.cf /etc/mail/sendmail.cf.old
# cp sendmail.cf /etc/mail/sendmail.cf
# /etc/init.d/sendmail restart
```

Camram is now integrated into your Sendmail installation for all users on the system.

PROCMAIL INTEGRATION

The code below illustrates a procmail recipe showing Camram integration. This can be specified on a per-user basis by placing the recipe in each user's .procmailrc file or in a system-wide /etc/procmailrc file.

```
MAILDIR=$HOME/Maildir
DEFAULT=$MAILDIR/
ORGMAIL=$MAILDIR/
# Directory for storing procmail configuration and log files
PMDIR=/var/log/procmail
# Put ## before LOGFILE if you want no logging (not recommended)
LOGFILE=$PMDIR/log
# Set to yes when debugging
VERBOSE=no
# Remove ## when debugging; set to no if you want minimal logging
## LOGABSTRACT=all
# Replace $HOME/Msgs with your message directory
# Mutt and elm use $HOME/Mail
# Pine uses $HOME/mail
# Netscape Messenger uses $HOME/nsmail
# Some NNTP clients, such as slrn & nn, use $HOME/News
# Mailboxes in maildir format are often put in $HOME/Maildir
#MAILDIR=/var/spool/spamtrap      # Make sure this directory exists!
##INCLUDERC=$PMDIR/testing.rc
##INCLUDERC=$PMDIR/lists.rc
:0fw
| /usr/local/camram/bin/procmail_filter
:0
* < 2
/dev/null
```

If you are not using Maildir-formatted mailboxes, you should change the lines that read

```
DEFAULT=$MAILDIR/
ORGMAIL=$MAILDIR/
```

to be

```
DEFAULT=
ORGMAIL=
```

Camram Configuration

Besides the procmail recipe, Camram has three files that can be changed to adjust its behavior:

- /usr/local/camram/ancillary/global_configuration—Default values; we do not make any changes to this file
- /var/spool/camram/configuration—Where most site-specific changes are made to adjust Camram's functions
- /usr/local/camram/ancillary/camram.local—The email relay script used to control the parameters when sending messages from Camram

We also cover how to set up appropriate cron jobs and Camram users at the end of this section.

/VAR/SPOOL/CAMRAM/CONFIGURATION

The valid parameters in the configuration file are the same ones that are valid in the global_configuration file. The configuration file is broken down into the following sections:

Core
Spam analysis
Spam storage
Filter configuration
User email addresses

All of the changes we list next are confined to the Core section. Besides the ones we cover here, some of the parameters you should consider adjusting include any keyword involving a path or any of the CRM114-scoring thresholds. A default file with just the section headers (listed previously) is created at Camram install time. You might want to make a copy of this file before making changes to it. At a minimum, the following parameters should be defined under the Core section in order to change from their default values:

```
authorized_users = comma-list:root,esj,dale
```

This should be a comma-separated list of privileged users who can manage the server via the GUI.

```
challenge_URL_base=string:http://mydomain.com/camram/pdgen.cgi
```

This is the parameter indicating the URL address for the challenge Web page. Change "mydomain.com" to be the address of your Web server.

```
correction_URL =string:http://mydomain.com/camram/correct.cgi
```

This is the URL where users enter corrections for messages misclassified as spam or ham.

```
reinjection_SMTP_port = string:30025
```

This is the port where Camram sends messages back to the MTA. If you used our example, leave this at 30025.

```
central_administration = boolean:0
```

This controls whether end users have access to the CRM114 retraining (0) or only the administrator has access to retraining (1). We recommend setting this to 0 so that end users can train their own filters.

```
password_key=string:notswordfish
```

This is the key used for the private password mechanism. Be sure to change it!

```
log_level=integer:1
```

The default logging level is 1. This value can be anything from 0 to 9, where 0 is no output and 9 is very verbose. Unless you are troubleshooting a problem, 1 should be acceptable. Messages are logged in /var/log/messages.

/USR/LOCAL/CAMRAM/ANCILLARY/CAMRAM.LOCAL

The camram.local file is the script that starts up the email forwarder program, EmailRelay. A few changes need to be made in this file, but before going through those, be sure to make a copy of the file as it was initially distributed:

```
# cd /usr/local/camram/ancillary
# cp -p camram.local camram.local.orig
```

This makes a backup copy of camram.local as camram.local.orig. This script is automatically read each time Camram is run, so there is no need to perform any steps to make changes to this file active. You should consider making the following changes to the parameters in this file:

```
camram_architecture=procmail
```

If you are running Camram on the same machine as the email boxes (as we are in our example), this should be procmail; otherwise, it should be set to intercept.

```
stamper_interface=ip address
```

This is the IP address of the interface that stamped messages should be accepted on. ip address should be set to the internal IP address, which accepts email from users on your local network. Do not set this to any externally available IP address or you could stamp messages for spammers!

```
filter_interface=ip address
```

This defines the interface of the server where email from the Internet originates. ip address should be set to an externally accessible interface that the MX record for your domain is set to, or a host that accepts mail for your domain.

```
local_smart_host=ip address
```

This is the machine that knows how to route email from your server/domain or if your Camram machine is behind a firewall. If your Camram machine is the smart host gateway, then set this to 127.0.0.1.

After changes are made to this file, the script must be invoked.

Executing the script /etc/init.d/camram.local start will start the script on many Linux systems.

CRON JOBS

There are three cron jobs that need to be set up to perform various tasks. Camram distributes suggested cron entries for each.

/usr/local/camram/ancillary/sweepup.py

This script deletes messages in the Camram dumpster. Be sure to run this often enough so that directory lookups don't get too slow when too many files are present.

/usr/local/camram/ancillary/clean_mail_queue.py

This script forwards feedback to the end user and should be run very often. Camram's example cron job runs every minute.

/usr/local/camram/ancillary/mbox2spamtrap.py

This script automatically scans the missed_spam_box folder for messages incorrectly classified by CRM114 and retrains it accordingly.

SETTING UP CAMRAM USERS

Each user who is going to have email (we assume all users) will need to have the appropriate directory and files set up on the system. This is accomplished by running the following script for every user on the system:

```
# /usr/local/camram/ancillary/clean_configuration.py -u username
```

You need to change username to be the user you want to set up on the system. If you are running Camram in intercept mode (without delivery to mailboxes on the machine Camram is running on), you need to create those accounts with the following command:

```
# /usr/local/camram/ancillary/new_account.py -u username
```

If you are running in procmail mode (like we are in our example), the accounts already exist as "real" UNIX users, so this step must be skipped.

After setting up your users, you must run the edit_config_cgi script to create user accounts and set up the database by going to the URL http://mydomain .com/camram/edit_config.cgi. Change "mydomain.com" to be the name of the Camram server you set previously. See the next section for using this screen.

Using Camram

Camram classifies messages into three possible categories:

- Red—Definitely spam; delivered to junk email folder
- Yellow—Possibly spam, possibly non-spam; delivered to spamtrap (possibly spam) folder
- Green—Definitely not spam; delivered to inbox

This results in an accurate classification system of messages, and it requires users to look in their spamtrap folder. Camram has a Web interface for accessing many functions. If you have followed our previous examples, the following screens are available at the listed URLs.

http://mydomain.com/camram/edit_config.cgi

The edit_config screen allows the administrator to edit the default settings for each user and should be run after adding each user, to perform initial setup.

http://mydomain.com/camram/correct.cgi

This screen allows the user to correct CRM114 misclassifications via a Web browser.

http://mydomain.com/camram/recover.cgi

The recover screen allows user access to the junk email folder (Camram calls it the dumpster) from a Web browser.

PREFERENCES

The parameters in the edit_prefs.cgi screen are stored in each user's home directory, in ~user/.camram/configuration. However, the parameters should be changed only by the preferences Web interface and not directly via editing the file, as changes will likely get overwritten. The parameters listed here are the same ones that are defined by the global_configuration file shown previously.

The defaults here are reasonable. The field labeled my_email_addresses should be updated with all aliases for each user. These addresses represent the addresses for which Camram will accept Hashcash stamps for this account.

SPAMTRAP

The correct.cgi screen allows each user to manage their spamtrap (yellow messages), which is the mail folder containing messages that Camram was unsure about when it ran the classifier on them. Simply check the checkbox on the left side for each misclassified message and click the Process button, and your messages will be sent to your inbox and the Bayesian classifier will be retrained.

RECOVER

The recover.cgi screen lists all messages in the dumpster and inbox. This screen allows you to pull false positive messages out of the dumpster and into the spamtrap for reclassification. Copies of all messages processed and classified as spam or not spam will end up in the dumpster. Do not be alarmed by the presence of non-spam emails. It is unfortunate that the dumpster contains more than just rejected spam messages because you can't just browse it quickly to identify false positives.

THOMAS SLUYTER

getting what you want

THE FINE ART OF PROPOSAL WRITING

In daily life Thomas is part of a small yet highly flexible UNIX support department at ING Bank in the Netherlands. He took his first steps as a junior UNIX sysadmin in the year 2000. Thomas part-times as an Apple Macintosh evangelist and as a board member of the Dutch J-Pop Foundation.

tsluyter@xs4all.nl

THROUGHOUT THE LAST TWO YEARS I have written a number of technical proposals for my employer. These usually concerned either the acquisition of new hardware or modifications to our current infrastructure.

Strangely enough, my colleagues didn't always achieve the same amount of success with their proposals as I did, which got me to thinking, "How does one write a proper proposal anyway?"

This mini-tutorial aims to provide a rough outline of what a proposal should contain, along with a number of examples. Throughout the document you'll also find a number of Do's and Don'ts to point out common mistakes.

The examples predominantly focus on the acquisition of hardware. This is due to the nature of my line of work, but let me say that the stuff I'll be explaining applies to many other topics. You may just as well apply them to desired changes to your network, some software that you would like to use, and even to some half-assed move that you want to prevent management from making.

I've never been a great fan of war, but Sun Tzu really knew his stuff! Even today his philosophies on war and battle tactics are still valid and are regularly applied. Not just in the military—these days it's not uncommon to see corporate busybodies reading *The Art of War* while commuting to work. In between my stuff you'll find quotes from Sun Tzu I thought were applicable to the subject matter.

Zen and the Art of Getting What You Want

"Though we have heard of stupid haste in war, cleverness has never been seen associated with long delays."
Sun Tzu, *The Art of War*

It happens occasionally that I overhear my colleagues talking about one of their proposals. Sometimes the discussion centers on why their idea got shot down and the question, "What the heck was wrong with the proposal?" They had copied a proposal that had worked in the past and replaced some information with their own. When I ask them to show me said proposal, I'm presented with two sheets of paper, of which one is the quote from our vendor's sales department. The other consists of 30% header/footer, a short blurb on what we want to buy, and a big box repeating all of the pricing info.

The problem with such a document, of course, is that management gets its nose rubbed in the fact that we

want to spend their money (and loads of it, too). To them such a proposal consists of a lot of indecipherable technical mumbo-jumbo (being the quote and some technical stuff), with the rest of the document taking up money-Money-MONEY.

While to you it may seem that the four or five lines of explanation provide enough reason to buy the new hardware, to management this will simply not do.

In writing a proper proposal, it is essential to keep your organization's upper echelons in mind. However, don't forget about your colleagues, either. It is more than logical that you should run a proposition past your peers to see whether they agree with all of the technicalities.

In order to make sure that both your targets agree on your proposal, you will have to: (1) employ tech-speak to reach your peers; and (2) explain your reasoning in detail to your management.

To craft such a document, there are a number of standard pieces to the puzzle that you can put in place. I'll go over them one by one. One thing I want you to realize, though, is that drafting a proposal will take time. Expect to spend at least half a day on writing a modest proposal.

Pieces to the Puzzle

My proposals tend to consist of several sections, some of which are optional, as not every type of proposal requires the information contained therein. For instance, not every project will require resources that can easily be expressed in numbers, and hence there may be no need for a list of costs. Here is the canonical list of proposal sections:

- Summary—Describes briefly your current problem, your solution for this, and the estimated costs.
- Introduction/scope—Gives your audience a clear picture of the troubled environment involved.
- Problems and proposed solutions—Describes in detail what is wrong, what the repercussions are (and what they may become), and your proposed solution.
- What is required—A list of things that you'll need to fix the problem.
- Other options—Of course, management wants the ability to save money. Here's where you give them the option to do so.
- Making it work—Describes which departments need to put in resources and what their tasks will be.
- Breaking things down—The costs of the various options set off against their merits and flaws.
- Final words—A last plea to your audience.

Now let's detail them.

THE SUMMARY

"The art of war teaches us to rely not on the likelihood of the enemy's not coming, but on our own readiness to receive him." Sun Tzu, *The Art of War*

Keep this part as short and simple as possible. Use one, maybe two short paragraphs to describe the current situation or problem and describe how you'll fix it. Use very general terms and make sure that it is clear which of the reader's needs you are addressing.

Be very careful not to put too much stuff in this section. Its main purpose is to provide the reader with a quick overview of the problem you're trying to solve and your final goal. This allows the reader to grasp quickly the subject of your proposal and helps ensure that it will be found more easily on a cluttered desk. A short summary means quick recognition. Here is an example:

> In the past year UNIX Support have put a big effort into improving the stability and performance capacity of their BoKS and NIS+ infrastructural systems. However, the oldest parts of our infrastructure have always fallen outside the scope of these projects and have thus started showing signs of instability. This in turn may lead to bigger problems, ending in the complete inaccessibility of our UNIX environment.
>
> I propose that we upgrade these aging servers, thus preventing any possible stability issues. The current estimated cost of the project is $16,260.

INTRODUCTION/SCOPE

When it comes down to the technical nitty-gritty, most managers have only a broad view of things going on in the levels beneath them. That's the main reason why you should include a short introduction on the scope of your proposal.

Give a summary of the services that the infrastructure delivers to the "business." This helps management form a sense of its importance. If a certain service is crucial to your company's day-to-day operation, make sure that your reader knows this. If it will help paint a clearer picture, you can include a simple graphic on the infrastructure involved.

The whole point of this section is to imprint on management that you are trying to do something about their needs, not yours. It's one thing to supply you with resources to tickle one of your fancies, but it's a wholly different thing to pour money into something that they themselves need. Here's an example:

> BoKS provides our whole UNIX environment with mechanisms for user authentication and authorization. NIS+ provides all of the Sun Solaris systems from that same environment with directory services, containing information on user accounts, printers, home directories, and automated file transfer interfaces.
>
> Without either of these services it will be impossible for us to maintain proper user management. Also, users will be unable to log in to their servers should either of these services fail. This applies to all departments making use of UNIX servers, from Application and Infrastructure Support all the way through to the Dealing Room floor.

PROBLEMS AND PROPOSED SOLUTIONS

"Whoever is first in the field and awaits the coming of the enemy, will be fresh for the fight; whoever is second in the field and has to hasten to battle will arrive exhausted." Sun Tzu, *The Art of War*

When writing your proposal, try to keep others' perspectives in mind. Try to anticipate any questions your reader might have and introduce your ideas in a way that will appeal to your audience. If you simply describe your goal instead of providing proper motivation, you'll be the one who "is second in the field."

In the previous section of your document you provided a quick description of the environment involved. Now you'll have to describe what's wrong with the current situation and what kind of effects it may have in the future. If your proposal covers the acquisition or upgrade of multiple objects, cover them sepa-

rately. For each object, define its purpose in the scope you outlined in the previous section. Describe why you will need to change its current state and provide a lengthy description of what will happen if you do not.

However, don't be tempted to exaggerate or to fudge details so that things will seem worse than they are. First, a proposal that is overly negative may be received badly by your audience. Second, you will have to be able to prove all of the points you make. Not only will you look like an ass if you can't, but you may also be putting your job on the line! So try to find the middle road. Zen is all about balance, and the "art of getting what you want" should also be. Here's an example:

> Recently the master server has been under increased load, causing deterioration of both performance and stability. This in turn may lead to problems with BoKS and with NIS+, which most probably will lead to symptoms like: Users will need more time to log in to their UNIX accounts; users may become unable to log in to their UNIX accounts; user accounts and passwords may lose synchronicity.

Close off each subsection (one per object) with a clearly marked recommendation and a small table outlining the differences between the current and the desired situation. Keep your recommendation and the table rather generic. Do not specify any models or makes of hardware yet.

The example below is focused on upgrading a specific server, but you can use such a table to outline your recommendations regarding just about anything—versions of software, for example, or specifics regarding your network architecture. It will work for all kinds of proposals:

> UPGRADES: UNIX Support recommends upgrading the master server's hardware to match or exceed current demands on performance.
>
> | *Current* | *Recommended* | |
> |---|---|---|
> | *System type* | Sun Netra T1 200 | —— |
> | *Processor* | Ultrasparc IIe, 500MHz | 2x Ultrasparc IIIi, 1GHz |
> | *Memory* | 512MB | 1 or 2GB |
> | *Hard drives* | 2x 18GB + 2x 18GB ext. | 2x 36GB, int. mirror |

The point of this section of your proposal is to convince your readers that they're the captain of the Titanic and you're the person who can spot the iceberg in time. All is not lost. Yet.

WHAT IS REQUIRED TO MAKE THIS WORK?

"The general who wins a battle makes many calculations in his temple ere the battle is fought. The general who loses a battle makes but few calculations beforehand."
Sun Tzu, *The Art of War*

Now that you have painted your scenario, and you've provided a vision of how to go about solving things, you will need to provide an overview of what you will be needing.

Don't just cover the hardware you'll need to acquire, but also take the time to point out which software you'll need and, more important, which departments will need to provide resources to implement your proposal. Of course, when it comes to guesstimates regarding time frames, you are allowed some slack. But try to keep your balance and provide your audience with an honest estimate.

One thing, though: Don't mention any figures on costs yet. You'll get to those later on. Here's an example of what it takes to get "it" to work:

A suitable solution for both Replica servers would be the Sun Fire V210. These systems will come with two Ultrasparc II processors and 2GB of RAM installed. This configuration provides more than enough processing power, but is actually cheaper than a lower spec'ed V120.

OTHER OPTIONS

"Do not interfere with an army that is returning home. When you surround an army, leave an outlet free. Do not press a desperate foe too hard." Sun Tzu, *The Art of War*

The above quote seems to be embodied in one of Dilbert's philosophies these days: "Always give management a choice between multiple options, even if there is only one."

Of course, in Dilbert's world, management will always choose the least desirable option, for instance choosing to call a new product the "Chlamydia," because "it sounds Roman." It will be your task to make the option *you* want to implement the most desirable in the eyes of your readers.

In case your proposal involves spending money, this is where you tell management: "All right, I know times are lean, so here are a number of other options. They're less suitable, but they'll get the job done." Be sure that even these alternatives will do the job you want them to. Never give management an option that will not be usable in real life. Here's an example:

> Technically speaking, it is possible to cut costs back a little by ordering two new servers instead of four, while re-using two older ones. This alternate scenario would cut the total costs back to about $8360, saving $3300.

If the main subject of your proposal is already the cheapest viable option, say so. Explain at length that you have painstakingly eked out every penny to come up with this proposal. Also mention that there are other options, but that they will cost more money/resources/whatever. Feel free to give some ballpark figures. Here's an example:

> Unfortunately, there are no cheaper alternatives for the Replica systems. The Sun Fire V120 might have been an option, were it not for the following facts:
>
> - It is not in the support matrix as defined by UNIX Support.
> - It is not natively capable of running IP Multi Pathing.
> - It will reach its so-called End of Life state this year.

Basically, you need to make management feel good about their decision to give you what you want. You really don't want them to pick any solution other than the one you're proposing, but you are also obliged to tell them about any other viable possibilities.

MAKING IT WORK

For some projects, you are going to need the help of other people. It doesn't matter whether they are colleagues, people from other departments, or external parties. In this section you will make a list of how many resources you are going to need from them.

Instead of going into heavy details, just give a broad description of the tasks laid out for these other parties. Estimate how many hours it will take to perform these tasks and how many people you will need from each source. Such a list will not only give management a clear picture of all of your necessities, but will also provide your readers with the scale of the whole project. Here's an example:

In order to implement the proposed changes to our overall security we will require the cooperation of a number of our peer departments:

- Information Risk Management (IRM) will need to provide AS and our customers with clear guidelines, describing the access protocols which will be allowed in the future. It is estimated that one person will need about 36 hours to handle all of the paperwork.
- Security Operations (GSO) will need to slightly modify their procedures and some of the elements of their administrative tools to accommodate the stricter security guidelines. It is estimated that one person will need about 25 hours to make the required alterations.

Breaking Things Down

You'll need to make this section as short as possible, since it covers the costs of all of the viable options that you provided in previous sections. Create a small table, setting off each option against the costs involved. Add a number of columns with simple flags you can use to steer the reader to the option of your choice.

Perhaps it will help clarify to recall product comparison charts in consumer magazines or sites on the Web. In comparing products they often include a number of columns marked with symbols like + (satisfactory), ++ (exceeds expectations), – (not too good), or – – (horrific). This will allow you to compare a number of distinct qualities in the options before you.

It goes without saying that you should be honest when assigning these values. If another option starts to look more desirable by now, you really have to re-evaluate your proposal. For example:

System	*Hardware*	*Costs**	*Current Needs*	*Expected Growth*
Master	V240, 2x CPU, 2GB RAM, 2x 36GB HD, 1x DVD	$6,638.10	+	+
Master-alt	V480, 2x CPU, 4GB RAM, 2x 36GB HD, 1x DVD	$20,575.00	++	++
Replica	V210, 2x CPU, 2GB RAM, 2x 36GB HD, 1x NIC	$4,811.10	++	++

*Price per unit. Multiply by amount of systems to get full price.

FINAL WORDS

"The clever combatant imposes his will on the enemy, but does not allow the enemy's will to be imposed on him." Sun Tzu, *The Art of War*

Use two or three concluding paragraphs to impress your reader with the force of your arguments. Shortly summarize the change(s) that you're proposing and repeat your arguments. Be firm, yet understanding. Here's an example:

> We have provided you with a number of possible scenarios for replacement, some options more desirable than others. In the end, however, we are adamant that replacement of these systems is necessary and that postponing these actions may lead to serious problems within our UNIX environment, and thus in our line of business.

Regarding Tone and Use of Language

Keep in mind at all times who your target audience is. It is quite easy to fall back into your daily speech patterns when writing an extensive document; at some point that may lead to catastrophe.

Assume that it is all right to use daily speech patterns in a document that will not pass further than one tier above your level (meaning your supervisor and

your colleagues). However, once you start moving beyond that level you will really need to tone it down.

Some points of advice:

- Avoid expletives at all times.
- Avoid using technical slang. Of course you're free to use standard technical terms, but leave out terms such as "boxen," "a win" (or "winnitude," for that matter), and "kludge."
- Avoid overly long sentences. This is a trap I fall into quite easily myself, as you may have noticed while reading this article. It tends to make it difficult to follow a train of thought.
- Don't be afraid to use your vocabulary. Words like "adamant" and "imperative" tend to have more impact than "I'm very sure" or "It is important."

Regarding Versioning and Revisioning

At ING Bank we include a small table at the beginning of each document which outlines all of the versions that document has gone through. It shows when each version was written and by whom. It also gives a one-liner regarding the modifications, and each version has a separate line showing who reviewed the document.

Of course, it may be wise for you to use different tables at times: one table for versions that you pass between yourself and your colleagues and one for the copies that you hand out to management. Be sure to include a line for the review performed by your supervisor in both tables. It's an important step in the life cycle of your proposal.

This may be taking things a bit far for you, but it's something we've grown accustomed to.

Final Thoughts

"Begin by seizing something which your opponent holds dear; then he will be amenable to your will." Sun Tzu, *The Art of War*

In other words, management is almost sure to give in if you simply make sure they know things will go horribly wrong if you are not allowed to do what you just proposed.

Of course, no method is the be-all and end-all of writing proposals, and mine is no exception. Some may simply find it too elaborate, while in other cases management may not be very susceptible to this approach. Try to find your own middle road between effort and yield. Just be sure to take your time and be prepared for any questions you may get about your proposal.

MARK BURGESS

the profession of system administration

OSLO UNIVERSITY OFFERS MASTER'S DEGREE IN NETWORK/SYSTEM ADMINISTRATION

Mark Burgess has a Ph.D. in theoretical physics and lives in Oslo, Norway. A 14-year veteran of system administration research, he has written 9 books and has published over 40 papers in international journals.

Mark.Burgess@iu.hio.no

Editor's Note: For system administration to be recognized as a profession, some sort of accredited training must exist. I asked Mark to write about his university's new program as an FYI for our readers and a spur for other universities.

Mark was recently made the world's first full professor specifically in the field of system administration. His paper "On the Theory of System Administration," published in *Science of Computer Programming* last year, has been the journal's top downloaded paper since it appeared.—RK

AT OSLO UNIVERSITY COLLEGE, THE two-year international Master's degree in Network and System Administration has now been running for just over a year. All teaching is carried out in English.

Although there has been no advertising of the course, the number of applications has been high. In the first year, 99 people applied for the 10 places, with 32 applications coming from outside Norway. This year, 155 applied for the 10 places, of whom 55 were from outside Norway. Many potential students apply from from Asia and Africa, and a few from Europe. Only one so far has applied from the U.S.

Students are led through four stages of development in the four semesters of the course:

- Learning basic principles
- Developing independent investigative skills and learning how to write documentation
- Extending analytical and critical skills
- Putting everything together in a final project, preferably in an industrial setting

Some unusual aspects of the curriculum include a course in supercomputers and virtual machines (with the generous help of IBM); a course in analytical methods based on Mark Burgess's latest book, *Analytical Network and System Administration: Managing Human-Computer Systems;* and a comprehensive course in ethics and social aspects of system administration.

ADAM TUROFF

practical perl

ERROR HANDLING PATTERNS IN PERL

Adam is a consultant who specializes in using Perl to manage big data. He is a long-time Perl Monger, a technical editor for *The Perl Review*, and a frequent presenter at Perl conferences.

ziggy@panix.com

HANDLING ERRORS IS THE BANE OF any program. In some programming languages, error handling is tedious to do properly, so we often forget to do it, or we do it improperly. In Perl, there are a few common idioms for handling errors that are both robust and easy to use.

I'm a big fan of design patterns in software development. Through patterns, we can talk intelligently about the code we write, the modules we use, and the behavior (and misbehavior!) of the software we come across on a regular basis. Patterns are a vocabulary that lets us focus on the big picture and ignore the meaningless high-level or low-level details.

In thinking about software design patterns, many people reach for the book *Design Patterns: Elements of Reusable Object Oriented Software*, written by Erich Gamma et al. However, patterns are a deep topic, and there is much more to know about patterns than is found in that book.

The concepts behind patterns did not start with one book describing better ways to build object-oriented software. In fact, the idea started with an alternative view of architecture put forth by Christopher Alexander in the 1970s. Alexander's premise was that we need a common language to discuss architectural principles—something the customer, the engineer, and the architect can all understand. When specialists focus on minutiae or elevate professional fashion over customer needs, we all lose.

Alexander's key insight is that we can work together to build open-ended vocabularies that describe the systems we build—whether they are buildings, towns, cars, airports, compilers, network monitoring software, or Web-based applications. In the realm of software, patterns are about describing the behavior of a module of code or an entire system. Once you start to see a pattern, it is easy to see the pattern repeated. From there, it is easier to repeat the good patterns and avoid the bad ones.

Patterns in Perl

Patterns came to software development through analysis of object-oriented systems. A classic pattern describes how to construct an object with a specific set of known behaviors, and how to combine compatible objects based on the patterns they implement. This school of design is quite prevalent within the Java community. If you have ever come across an iterator or a factory, you've seen some of the behaviors described in *Design Patterns* in use.

But patterns are not restricted to objects or any other domain. Because patterns are an open-ended vocabulary, we can use patterns to describe different levels of software, ranging from a single line of code all the way up to a large, complex project like a database server or a Web-based content management system. Patterns are *everywhere* in software.

For a concrete example, look at the following patterns for error handling. If you are familiar with C, you may have seen this idiom for opening a file:

```
FILE *fp;
fp = fopen("/my/file", "r");
if (fp == NULL) {
   return -1;  // ERROR - could not open file
}
```

In Perl, there's always more than one way to do it. If you learned how to program in C, you can program in Perl in a C-like manner:

```
open(my $fh "/my/file");
if (!$fh) {
   return -1;  ## ERROR - could not open file
}
```

In these brief snippets, there is exactly one operation being performed: opening a file. If there is any problem opening this file, then the operation terminates immediately.

Here is a more natural expression of the same basic intent in Perl:

```
open(my $fh "/my/file") or return -1;
```

In this formulation, there is one operation to perform, and it is expressed all at once in a single statement. Furthermore, the intent of the whole statement reads quite naturally: *do something or recover immediately*. Not only is this statement easier to write, but it is much easier to read. Consequently, this statement is also easier to remember and get right the first time.

In many simple scripts, it is common or even advisable to terminate immediately at the first point of failure. This pattern is known as "open or die," and it is one of the most common patterns in Perl programming:

```
open(my $fh "/my/file") or die "Cannot open '/my/file'\n";
```

Using "open or die" may seem extreme at first, but it provides a simple way to express a set of necessary preconditions for a script. For example, consider a script run periodically by cron that needs to read and write some files. If any of those files are missing or cannot be created, the script cannot proceed. If it does continue to run, it could generate bad output, or, in the worst case, do damage to a running system. Using the "open or die" pattern allows this script to open all of its files and succeed, or gracefully terminate when any one of its files cannot be opened.

Error Handling in Perl

How does the "open or die" pattern work? The key is the ultra-low-precedence or operator that connects the two statements. If there is any true value whatsoever on the left half of the expression (in this case, the result of an open operation), it will return that value immediately and not evaluate the right-hand side (die). The right-hand side (die) will be executed only if the left-hand side (open) returns a false value.

This pattern uses the or operator instead of the higher-precedence || (Boolean or) operator, for a couple of reasons. First, it is clearer when reading the code. Second, because or is an ultra-low-precedence operator, there is no ambiguity:

```
## The first statement *must* be "open" with two parameters
## The second statement *must* be "die" with one parameter
open FH, "/my/file" or die "Cannot open '/my/file'\n";
```

Using the higher-precedence || operator would be ambiguous to Perl:

```
## Is the second parameter "/my/file" || ....
## or is it an open followed by || die?
open FH, "/my/file" || die "Cannot open '/my/file'\n";
```

Another important characteristic of "open or die" is the behavior of the left-hand side of the expression, open. If open encounters *any* error whatsoever, it will return a false value. For any other result, it will return *some* true value. Therefore, the die clause in this statement will execute only when there is an open failure. (The actual cause of the failure can be found elsewhere, in the special variable $!.)

The true power in this small pattern is not that it is a concise expression of the proper behavior for opening files, but in its general utility in similar contexts. Most Perl functions that handle system interaction provide the same behavior—return false on error, true on success. This includes functions like chdir, mkdir, unlink, and so on. For conciseness, Perl programmers generally call this overall pattern "open or die."

In C, the pattern is just the opposite—return zero (false) on success, and a non-zero error code (true) on error. This leads to cumbersome idioms like the example above with fopen. In Perl, the system built-in function behaves in a C-like manner, returning false (zero) on success, and a true (non-zero) value on failure. The result is similar to the cumbersome "system and die" pattern, since the system built-in adheres to this C-like behavior:

```
system "/bin/date" and die "Can't find '/bin/date'";
```

(The "system and die" pattern is generally regarded as broken. Larry Wall has declared that this unfortunate misfeature will be fixed in Perl 6.)

Recovering from Errors in Perl

Using the "open or die" pattern is a great way to terminate your script at the first sign of error. But sometimes you do not want to terminate your script. Rather, you need to exit immediately from a subroutine, break out of a loop, or just display a warning message.

Even though this pattern is commonly called "open or die," the right-hand side doesn't need to call die. Any recovery action fits the pattern, including return, warn, or even print.

Below is a sub that takes a filename and returns all non-blank lines that do not start with a hash character (i.e., comments). If it cannot open a file, it exits immediately and returns false:

```
sub get_lines {
   my $filename = shift;
   my @lines;

   open(my $in, $filename) or return;
   while (<$in>) {
      next if m/^$/;  ## Skip blank lines
      next if m/^#/;  ## Skip comment lines
      push (@lines, $_);
   }

   return @lines;
}
```

Carp, Croak, and Friends

Functions like die and warn can report exactly where an error occurred. That may work for scripts, where the cause of the error is likely nearby. But this behavior does not work very well when using modules. Although your program may have issued a warning or terminated at line 135 of SomeModule.pm, that message may not mean anything to you, especially if you installed SomeModule from CPAN.

It makes more sense to identify the location of the code that led to the error in SomeModule.pm. This is more likely the cause of the problem, especially when using well-tested modules. This is the problem that the Carp module solves. Carp is a standard module that is bundled with Perl which provides the error-reporting functions carp and croak, which can be used in place of warn and die. When these functions display a warning or a termination message, they report the location where the current sub was called, not the location where an error was encountered (i.e., the location of the call to carp or croak).

Here is a simple program that demonstrates the difference between these two sets of functions:

```
1: package Testing;
2: use Carp;
3:
4: sub test_carp {
5:     carp "Testing carp";
6: }
7:
8: sub test_warn {
9:     warn "Testing warn";
10: }
11:
12: package main;
13:
14: Testing::test_carp();
15: Testing::test_warn();
```

And here is the result:

```
Testing carp at test.pl line 14
Testing warn at test.pl line 9
```

Note that carp focuses attention on where the sub test_carp was called, while warn focuses attention within the body of test_warn. This is why module authors should prefer the functions provided by Carp over the standard built-in functions.

Returning Errors in Perl

Patterns for handling errors are important. By understanding when and how functions like open return errors, we can use a clear and concise pattern for handling errors when they occur. The beauty behind this pattern lies in its extensibility. Not only can it be used with many recovery strategies, but it can be used with any sub that signals errors the same way open does.

Recall for a moment how open communicates errors: It returns a false value on failure, and some true value on success; any error message will be returned through a pre-defined variable ($! in this case). Any other sub that behaves in this manner can be used with the "open or die" pattern.

This process sounds simple enough, except that there is some subtlety involved. Remember that there are precisely five false values in Perl:

- The number 0
- The string "0"
- The empty string
- The empty list
- The undefined value (undef)

There is another, subtle wrinkle in Perl behavior. Subroutines can be called in one of two possible ways: in scalar context and in list context. In list context, there is precisely one false value, the empty list. All other values produce a list of one element, which is true. The following program illustrates the differences:

```
sub return_empty  {return;}
sub return_string {return "";}
sub return_zero   {return "0";}
sub return_0      {return 0;}
sub return_undef  {return undef;}

## Test scalar return values
($_ = return_empty) and print "True (scalar empty)";
($_ = return_string) and print "True (scalar string)";
($_ = return_zero) and print "True (scalar zero)";
($_ = return_0) and print "True (scalar 0)";
($_ = return_undef) and print "True (scalar undef)";

## Test list return values
(@_ = return_empty) and print "True (list empty)";
(@_ = return_string) and print "True (list string)";
(@_ = return_zero) and print "True (list zero)";
(@_ = return_0) and print "True (list 0)";
(@_ = return_undef) and print "True (list undef)";
```

As described above, this program produces the following output:

```
True (list string)
True (list zero)
True (list 0)
True (list undef)
```

These rules may sound complicated, but they really aren't. They help Perl do the right thing in a variety of common circumstances. This program demonstrates that if you want to return a false value in all circumstances, just use a simple return statement—it will return false whether you, the caller, uses list or scalar context. Therefore, any sub that uses this behavior can plug right into the "open or die" pattern with no extra effort.

Alternative Error Mechanisms in Perl

Returning a false value is often sufficient for signaling an error. But sometimes there are legitimate values returned that happen to be false but do not signal an error. Consider a sub that returns a series of numbers that could include zero, or a sub that returns a series of strings which could include the empty string. In these situations, it may not be feasible to say that "any false value" signals an error. In these cases, it is generally better to say that "the undefined value" signals an error.

A common example of this pattern is reading lines from a file:

```
while (<>) {
  …
}
```

The purpose of looping over a file line by line is to process one line at a time. However, sometimes it is possible to read an empty string from a file, or a line containing the single character 0. These values should not signal end-of-loop.

What is actually happening here is that Perl sees that construct and interprets it as this instead:

```
while (defined($_ = <>)) {
  …
}
```

This behavior allows Perl to act as we expect it should. The construct will read every line from the file, including blank lines and lines that contain the number zero. The undefined value will be returned when there is an error reading from the file, such as when trying to read past the end-of-file. Thus, Perl can read each and every line from a file (regardless of whether that line is "true" or not) and terminate when reaching end of file.

You can reuse this pattern in your programs as well. If you need to return false values (like zero) from a sub but still want to plug into the "open or die" pattern, just remember to check to see whether the return value is defined. If it is not, then some error must have occurred:

```
defined(add_user()) or die "Cannot add user";
```

Conclusion

Handling errors is a key aspect of any program. In Perl, there are many patterns for handling errors. If you are comfortable programming in a C-like manner, you can use the error-handling patterns that feel comfortable to you. However, native Perl patterns for handling errors are simpler to use and easier to get right the first time.

Corrections

In my last column on Class::DBI, I used this idiom to edit a temporary file:

```
sub get_input {
  open (my $fh, ">/tmp/library.data.$$");
  ….
}
```

Jeremy Mates wrote in, identifying this as a security flaw. I want to thank him profusely for pointing this out. Jeremy goes on to say:

> Insecure temporary file handling problems are unfortunately far too common in code still being written and used, despite being trivial to eradicate through the use of secure alternatives such as mktemp(1) and various modules in Perl, such as File::Temp.
>
> For soliciting input from an external editor, I recommend the use of my Term::CallEditor module, which uses File::Temp to create a secure temporary file that an editor can be run on.
>
> Thanks, Jeremy.

CLIF FLYNT

the tclsh spot

CREATING STAND-ALONE EXECUTABLES WITH TCL/TK

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: A Developer's Guide* and the TclTutor instruction package. He has been programming computers since 1970 and a Tcl advocate since 1994.

clif@cflynt.com

DYNAMIC LANGUAGES LIKE TCL ARE great for rapid development. In a few hours you can churn out applications that would take days or weeks (or even months) to develop in compiled languages like C, C++, or Java.

The downside of developing an application in a high-level dynamic language is that clients need to have the appropriate interpreters and libraries installed on their system before they can run your application.

The solution to this problem is to wrap the application and interpreter into a single executable. For Tcl, there are several choices:

- Tcl-Wrapper (http://sourceforge.net/projects/tclpro/): The first Tcl wrapper, it was developed by Scriptics as part of the TclPro development suite. This is the only wrapping application that stores the Tcl code as compiled bytecodes, providing some code obfuscation. This application is now supported by ActiveState (http://www.activestate.com) as part of the TclDev package.
- Wrap (http://www.xs4all.nl/~nijtmans/wrap.html): Jan Nijtmans created a proof-of-concept wrapping application to demonstrate how a smaller executable could be made using upx and wrap. Jan is no longer supporting this application, but it spawned the current StarPack, Freewrap, and TOBE wrappers.
- FreeWrap (http://freewrap.sourceforge.net/): This application is written by Dennis LaBelle and is based on D. Richard Hipp's mkTclApp. This is excellent for pure Tcl applications. The command-line interface is very simple and clean.
- StarPack (http://www.equi4.com/): Developed by Jean-Claude Wippler and Steve Landers, this is part of a set of deployment solutions that includes a single-file Tcl/Tk interpreter, compressed applications to be run by that single-file interpreter, and wrapped executables with interpreter, application code, and data. This package has more features than FreeWrap. It is very useful for wrapping a pure Tcl application or one that includes a Tcl-stubs-enabled library. It uses a more complex application build sequence, which may take a few steps to create an application.
- TOBE (http://www.hwaci.com), developed by D. Richard Hipp, provides the most control and supports extensions that are not Tcl-stubs-enabled. Using this package requires compiling a small "C" code wrapper and linking that to the Tcl libraries.

All of these wrapping solutions are built around a zip archive. Part of the zip-file specification allows a prefix to be placed ahead of the actual archive. The prefix can even be an executable program, which allows a zip archive to be an application. This is how self-extracting zip files are created. The Tcl wrapping programs all use a modified tclsh or wish interpreter as the prefix.

The problem with just prepending the tclsh interpreter to a zip file is that a useful Tcl interpreter is not just a single executable file. When the Tcl interpreter starts it loads a number of Tcl files with support for other commands.

In order to make a single executable, these files need to be included with the archive, and the Tcl interpreter needs to understand how to find the files.

Enter Tcl's Virtual File System (VFS) API to solve the problem. Just as UNIX streams generalized the interface between different devices, Tcl's VFS generalizes the interface between different directory systems. With a little bit of glue to read a directory format, any collection of files can be mounted as a directory and the files can be read. If the collection supports writing, they can also be written.

For example, the VFS API enables a Tcl script to mount an FTP site as a directory, search the site with the glob and cd commands, and open and read files with open, gets, and read commands.

For a wrapped application, this means that the Tcl support libraries can be placed in the zip archive and the Tcl interpreter can be told to look for them there, instead of looking for them on the hard drive (/usr/local/lib/tcl8.4, for instance).

The default search path for the Tcl libraries is compiled into the Tcl interpreter. You can define an alternative path to the libraries with the environment variables TCL_LIBRARY and TK_LIBRARY.

The basic steps in creating a wrappable Tcl interpreter are:

- Write glue functions to interface between the Tcl VFS and the file collection format.
- Write a function that will:
  1. Mount the file collection.
  2. Set TCL_LIBRARY to point to the new directory.
  3. Set TK_LIBRARY to point to the new directory.
  4. Initialize Tcl and Tk interpreters.
  5. Load and evaluate the Tcl application.
- Add code to invoke your Tcl initialization procedure.

Any of the wrappers described above will work for a pure Tcl application. To wrap a FORTRAN application, however, we need more control. The big problem is that a FORTRAN application will have a main entry point defined by the FORTRAN compiler/linker. This will conflict with the main function in a normal tclsh.

The TOBE paradigm lets us put the code to initialize the Tcl interpreter in a function that is invoked by the FORTRAN code, rather than in the normal main function.

When you download TOBE from http://www.hwaci.com/sw/tobe/index.html you get:

- Sample Makefile for Linux and cross-compiling with mingw.
- zvfs.c, the glue that lets a zip file be accessed like a file system.
- tkwinico.c, a function that provides a custom windows icon.

- main.c, the main function that mounts the zip file and initializes the Tcl interpreter.
- main.tcl, a sample Tcl application to wrap.

I'll start with a simple example of using TOBE to wrap a single script, and then show how to use TOBE to wrap a FORTRAN application.

The first step is to be certain you have Tcl and Tk static libraries available. Many distributions only include the dynamic libraries, but to make a completely self-contained executable, we need to link with static libraries.

If you don't have libtcl*.a on your system, you can build it.

To build Tcl from scratch:

1. Download the Tcl sources from http://sourceforge.net/projects/tcl/.
2. Untar the archives (tar -xvzof tcl8.4.6-src.tar.gz).
3. Change to the UNIX directory (cd tcl8.4.6/unix).
4. Configure the Makefile (./configure -disable-shared).
5. Make the libraries (make).
6. Repeat for tk.

Note that you need to include the -disable-shared option to configure. The default configure script makes shared libraries, but not static libs.

Also, you don't need to install the new libraries. We can build TOBE applications using a different version of Tcl than the default on our system.

The next step to use TOBE for a simple application is to edit the Makefile. The sample Makefile is created with hard links to Richard Hipp's home directories, and is guaranteed not to work for anyone else. However, the Makefile includes commented-out generic paths as examples of what might exist on your system.

Use your favorite editor to find each of the /drh/ lines in the Makefile and comment them out. Either uncomment a previous generic line or define a path that's appropriate to your system.

My preference is to place the TOBE and application directories in the same directory as the Tcl and Tk source directories, and to use relative paths in this format:

```
###### The linker option used to link against the TCL library
#
LIB_TCL = ../tcl8.4.6/unix/libtcl8.4.a -lm -ldl
```

The default Makefile has hooks for many Tcl extensions, including BLT, SQLite, and Img. The executable will be smaller if we don't include extensions we aren't using, so uncomment all of the -DWITHOUT_foo options in the Makefile.

Once this is done, you should be able to type make in the tobe directory, watch it build zvfs.o, main.o, etc., link these with the Tcl and Tk libraries, tack a little bit of zip magic onto the end of the file, and build an executable zip file.

Any errors indicate that you don't have the paths set correctly or are missing a -L option in a library path definition.

When this is done, you should be able to use unzip -t to examine the contents of the new file and confirm that it really is a zip file.

The sample main application in main.c is hardcoded to run the Tcl program main.tcl in the zip archive.

You can add code for a simple Tcl application to src/main.tcl, rerun make, and create a new tobe that will run that application. For real projects, modify the Makefile to generate an application with the name you prefer, or just rename the tobe executable this creates.

To use TOBE with the FORTRAN/Tcl library described in the previous couple of "Tclsh Spot" articles, we need to merge code from the TOBE main.c into ftcl_start (in ftcl_c.c) to initialize the interpreter from the zip archive, instead of using the default files located on your system.

The original ftcl_start function took a single argument, the name of the script to load. The modified version requires two arguments: the name of a script to load and the name of the executable. (Your FORTRAN program can get this information using the f2kcli package from http://www.winteracter.com/f2kcli/index.htm.)

We need to pass the name of the executable to Tcl_FindExecutable after creating the Tcl interpreter. The Tcl_FindExecutable function finds the full path to the application and saves it internally for use by a number of the Tcl interpreter's housekeeping tasks.

After creating the Tcl interpreter with

```
ftcl_interp = Tcl_CreateInterp();
Tcl_FindExecutable(executableName);
```

the code can set a few global variables. For this application, we aren't accepting any command-line flags, so the argv and argc global variables are set to empty and 0, respectively. The argv0 variable holds the name of the application, which is stored locally in an array named executableName. Finally, the tcl_interactive variable is set to false to let the Tcl interpreter know that it's running a script, not running as an interactive shell.

When tclsh is used as an interactive shell, the Tcl interpreter tries to evaluate each line as a Tcl command, and if that fails, it tries to evaluate the line as a system command. This is proper for an interactive shell, but inappropriate behavior for most scripts.

```
Tcl_SetVar(ftcl_interp, "argv", "", TCL_GLOBAL_ONLY);
Tcl_SetVar(ftcl_interp, "argc", "0", TCL_GLOBAL_ONLY);
Tcl_SetVar(ftcl_interp, "argv0", executableName,
        TCL_GLOBAL_ONLY);
Tcl_SetVar(ftcl_interp, "tcl_interactive", "0", TCL_GLOBAL_ONLY);
```

Next, the zip file system is mounted and Tcl's global environment array is set to point to the zip file system.

Tcl keeps a copy of the user's environment in the env array. All of the environment variables you can set in your shell are stored in this array, with the environment variable name used as the array index. For instance, puts $env(PATH) would print out the path.

In this case, since we need to force the Tcl interpreter to look for the initialization files in the zip archive, we overwrite the original values for the TCL_LIBRARY and TK_LIBRARY indices to point to the zip file system.

```
/* We have to initialize the virtual file system before calling
** Tcl_Init(). Otherwise, Tcl_Init() will not be able to find
** its startup script files.
*/

/* Initialize the zip file system package */
Zvfs_Init(ftcl_interp);

/* Mount the zip archive (this executable) as /zvfs */
retval = Zvfs_Mount(ftcl_interp, Tcl_GetNameOfExecutable(),
        "/zvfs");

/* Point env(TCL_LIBRARY) and env(TK_LIBRARY) to the zip direc-
        tories */
Tcl_SetVar2(ftcl_interp, "env", "TCL_LIBRARY", "/zvfs/tcl",
        TCL_GLOBAL_ONLY);
```

```
Tcl_SetVar2(ftcl_interp, "env", "TK_LIBRARY", "/zvfs/tk",
        TCL_GLOBAL_ONLY);
```

Now the code can initialize the Tcl and Tk interpreters with Tcl_Init and Tk_Init:

```
if( Tcl_Init(ftcl_interp) ) {
    ftcl_debug_message( "ftcl_start - Tcl_Init:",
    Tcl_GetVar(ftcl_interp, "errorInfo", TCL_GLOBAL_ONLY)) ;
    return 1;
}
if( Tk_Init(ftcl_interp) ) {
    ftcl_debug_message( "ftcl_start - Tk_Init:",
    Tcl_GetVar(ftcl_interp, "errorInfo", TCL_GLOBAL_ONLY)) ;
    return 1;
}
```

If your application might need to create new interpreters with extensions loaded, you must include a call to Tcl_StaticPackage to let the Tcl interpreter know that a statically linked package has been loaded into the interpreter.

For example, if you need to create child interpreters with Tk loaded, you might do it with these commands. Note that the load is invoked with an empty string and a package name, instead of the more common usage of providing a file name. This format is used when the file is already loaded and the script just needs to invoke the extension's initialization function in the new slave interpreter.

```
interp create withwish
withwish eval {load "" tk}
withwish eval {pack [canvas .c]}
withwish eval {.c create text 100 50 -text "child interp"}
```

If you leave out the Tcl_StaticPackage call, this code would generate an error message like

```
package "tk" isn't loaded statically
```

By including this line, the Tk package can be loaded into a new slave interpreter.

```
Tcl_StaticPackage(ftcl_interp,"Tk", Tk_Init, 0);
```

If your application has other extensions it may need to load into slave interpreters, you must include a call to Tcl_StaticPackage for each one.

We need to make a couple of modifications to the FORTRAN and Tcl code to run inside a TOBE application.

The code we had in lander.f90 called ftcl_start with the single argument config.tcl. The ftcl_start function then called Tcl_EvalFile to load and evaluate the config.tcl script in the current directory.

When we package and ship this application, we'll have a single file, and there won't be any config.tcl in the current directory. The config.tcl file will be in the zip archive, which is mounted as /zvfs.

Changing the original ftcl_start from this original:

```
CALL ftcl_start('config.tcl')
```

to this provides the application name and causes the Tcl_EvalFile to try to evaluate the file in the zip archive:

```
CHARACTER(256) :: exe
CALL get_command_argument(0,exe)
CALL ftcl_start('/zvfs/config.tcl', exe)
```

Similarly, in the config.tcl script that loads the GUI and starts the Lunar Lander application running, we originally just sourced files from the current directory like this:

```
source options.tcl
```

```
source setConditions.tcl
source GUI6.tcl
```

In order to run within a TOBE, we need to source these files from the /zvfs directory. The simple solution is to just add the /zvfs/ to the paths.

```
source /zvfs/options.tcl
source /zvfs/setConditions.tcl
source /zvfs/GUI6.tcl
```

The final steps are to compile the new ftcl_c.c, link the application, and create the magic TOBE zip archive.

These steps are all in the sample Makefile that comes with TOBE. They can be combined into a single sequence of commands like this:

```
lander: lander.f90 ftclz.a
    $(FC) -o tobe.zip $(FLAGS) lander.f90 ftclz.a $(LIBS)
    rm -rf zipdir
    mkdir zipdir
    ln -s /usr/local/lib/tcl8.4 zipdir/tcl
    ln -s /usr/local/lib/tk8.4 zipdir/tk
    cp $(TCL_APPFILES) zipdir
    cat ../tobe/src/null.zip >>tobe.zip
    cd zipdir; /usr/bin/zip -qr ../tobe.zip *
    cd ..
    mv tobe.zip lander
```

And with that, we've created a stand-alone Tcl/FORTRAN application that can be shipped to our client without worrying whether they have the proper Tcl interpreters installed on their system.

As usual, the complete code described in this article is available at http://www.noucorp.com.

GLEN MCCLUSKEY

working with C# serialization

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

glenm@glenmccl.com

AT SOME POINT IN THE DEVELOPMENT of most software applications, design decisions are made about how to store and retrieve application data. For example, if your application reads and writes to disk files, you need to make basic choices about how to represent the data on disk.

In this column we want to look a bit at C# I/O issues, and in particular at a mechanism called serialization. Serialization is used to convert C# objects into bytestreams, in a standardized way, so that those objects can be saved to disk or sent across a network.

The Need for Serialization

Let's start by considering a couple of examples. The first one writes a floating-point value to a text file and then reads it back:

```
using System;
using System.IO;

public class SerialDemo1 {
   public static void Main() {

      // write double value to text file

      double d1 = 0.1 + 0.1 + 0.1;
      StreamWriter sw =
            new StreamWriter("out", false);
      sw.WriteLine(d1);
      sw.Close();

      // read double value back from text file

      StreamReader sr = new
            StreamReader("out");
      string ln = sr.ReadLine();
      double d2 = Double.Parse(ln);
      sr.Close();

      // compare values

      if (d1 != d2) {
            Console.WriteLine("d1 != d2");
            Console.WriteLine("difference = " +
               (d1 - d2));
      }
   }
}
```

When this program is run, the result is:

```
d1 != d2
difference = 5.55111512312578E-17
```

For some reason, our attempt to store a floating value in a text file fails. If we know much about floating-point, we may not be surprised, given that many decimal values have no exact representation in binary. For example, the common value 0.1 is the sum of an infi-

nite series of binary fractions. Somehow our initial value got changed a bit, due to roundoff factors and so forth.

Here's another example. This time we're trying to store short (16-bit) values, and our program is as follows:

```
using System;
using System.IO;

    public class SerialDemo2 {
        public static void Main() {

            // write short value to binary file

            short s1 = 12345;
            FileStream fs1 =
                new FileStream("out", FileMode.Create);
            fs1.WriteByte((byte)((s1 >> 8) & 0xff));
            fs1.WriteByte((byte)(s1 & 0xff));
            fs1.Close();

            // read short value from binary file

            FileStream fs2 =
                new FileStream("out", FileMode.Open);
            int b1 = fs2.ReadByte();
            int b2 = fs2.ReadByte();
            short s2 = (short)((b2 << 8) | b1);
            fs2.Close();

            // compare short values

            if (s1 != s2) {
                Console.WriteLine("s1 != s2");
                Console.WriteLine("difference = " +
                    (s1 - s2));
        }
    }
}
```

Our approach in this code is to pick apart the values, write the individual bytes to a binary file, then read them back.

The output of this program is:

```
s1 != s2
difference = -2295
```

Unfortunately, we made a mistake when we read the value back from the file—we got the byte order wrong.

These examples serve to illustrate why serialization is important—there needs to be some standard mechanism for converting objects into bytestreams and back again.

A Serialization Example

Let's go back to the first example, where we are trying to store a floating-point value. Let's assume that we have such a value represented in an object, and we want to store that object to a file and then later retrieve it. Here's some code that will do so:

```
using System;
using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.Runtime.Serialization.Formatters.Soap;

[Serializable]
```

```
public class MyObject {
    private double dvalue;

    public MyObject(double dvalue) {
        this.dvalue = dvalue;
    }

    public double GetValue() {
        return dvalue;
    }
}

public class SerialDemo3 {
    public static void Main() {

            // serialize MyObject instance to file

            double d1 = 0.1 + 0.1 + 0.1;
            MyObject obj1 = new MyObject(d1);
            FileStream fs1 =
                    new FileStream("out", FileMode.Create);
            BinaryFormatter fmt1 = new BinaryFormatter();
            fmt1.Serialize(fs1, obj1);
            fs1.Close();

            // deserialize MyObject instance from file

            FileStream fs2 =
                    new FileStream("out", FileMode.Open);
            BinaryFormatter fmt2 = new BinaryFormatter();
            MyObject obj2 = (MyObject)fmt2.Deserialize(fs2);
            fs2.Close();
            double d2 = obj2.GetValue();

            // compare object values

            if (d1 != d2) {
                    Console.WriteLine("d1 != d2");
                    Console.WriteLine("difference = " +
                            (d1 - d2));
        }
    }
}
```

The example creates an instance of MyObject, containing a double value, and then creates a BinaryFormatter. The formatter is used to convert an object into a bytestream and takes care of all details of conversion, such as traversing arrays and object graphs (for example, linked lists).

At a later point, the process is reversed and the bytestream converts back into an object. Obviously, the serialization process is making use of some internal format for laying out objects and individual data items such as floating-point values.

Other kinds of formatting are possible. For example, if in our demo we go through and change BinaryFormatter to SoapFormatter, then the serialization format will be XML-based.

Using serialization in this way takes care of our original problem with being able to represent floating-point values exactly.

Transient Data

Making use of serialization is often as simple as the previous example illustrates, with all the details handled automatically. But it is also possible to exercise a finer degree of control over the process.

Let's consider another example, where we have an object containing an internal table, a table whose values are computed in some obvious way. By default, the serialization process will convert such tables into a bytestream, along with all other fields in the object. However, doing so may waste a lot of space.

To get around this problem, it is possible to specify that certain fields in an object not be serialized and, further, to specify a callback mechanism that is invoked when an object is deserialized. Using the callback, the object's table can be reconstructed at deserialization time.

Here's what the code looks like:

```
    using System;
    using System.IO;
    using System.Runtime.Serialization;
    using System.Runtime.Serialization.Formatters.Binary;

    [Serializable]
    public class MyObject : IDeserializationCallback {
        private uint length;
        private uint offset;
        [NonSerialized] private uint[] tab;

        private void buildtab() {
                tab = new uint[length];
                for (uint i = 0; i < length; i++)
                        tab[i] = i * i + offset;
    }

    public MyObject(uint length, uint offset) {
                this.length = length;
                this.offset = offset;
                buildtab();
    }

    public uint GetValue(uint index) {
                return tab[index];
    }

    public virtual void OnDeserialization(Object x) {
                buildtab();
    }
}

public class SerialDemo4 {
    public static void Main() {

            // serialize object to file

            MyObject obj1 = new MyObject(1000, 1000);
            FileStream fs1 =
                    new FileStream("out", FileMode.Create);
            BinaryFormatter fmt1 = new BinaryFormatter();
            fmt1.Serialize(fs1, obj1);
            fs1.Close();

            // deserialize object from file

            FileStream fs2 =
                    new FileStream("out", FileMode.Open);
            BinaryFormatter fmt2 = new BinaryFormatter();
            MyObject obj2 = (MyObject)fmt2.Deserialize(fs2);
            fs2.Close();
            Console.WriteLine(obj2.GetValue(25));
    }
}
```

The internal table in this example is a pretty simple one, a table of squares plus an offset. For example, the value for 25 will be 1625, or 25 * 25 + 1000. Since the table can be computed, there's no point in serializing it. Instead, we rely on the OnDeserialization method as a hook to be called when an instance of MyObject is deserialized.

In other words, only the table length and offset are serialized for MyObject instances, and these values are sufficient to reconstruct the object. The actual table is marked with a [NonSerialized] attribute. Doing it this way saves a great deal of disk space. More generally, you might wish to use a scheme of this sort when the physical and logical representations of an object are fundamentally different from each other.

Serialization is an important tool that you can use to store and transmit C# data in a standardized way. You no longer have to worry about devising custom data formats or about issues such as byte ordering.

PETER H. SALUS

the bookworm

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 owns neither a dog nor a cat.

peter@netpedant.com

BOOKS REVIEWED IN THIS COLUMN

INTRODUCTION TO COMPUTER SECURITY
Matt Bishop

Boston, MA: Addison-Wesley, 2004. Pp. 784. ISBN 0321247442.

THE MYTH OF HOMELAND SECURITY
Marcus J. Ranum

2nd ed., John Wiley & Sons, 2003. Pp. 240. ISBN 0471458791.

BUILDING APPLICATIONS WITH THE LINUX STANDARD BASE
Core Members of the Linux Standard Base Team

Upper Saddle River, NJ: Prentice Hall PTR 2004. Pp. 272 + CD-ROM. ISBN 0131456954.

LINUX JOURNAL 1994–2003 ARCHIVE

Seattle, WA: SSC, 2004. CD-ROM. (https://www.ssc.com/cgi-bin/lj/back_issue.) ISBN 1578310237.

THE CULT OF MAC
Leander Kahney

San Francisco: No Starch Press, 2004. Pp. 280. ISBN 1886411832.

I've got a lot of stuff to write about this month, not least because this will be my last "Bookworm." I have been writing reviews for *;login:* since 1989. That's 15 years. I have no idea how many books I've read, commented on, reviewed.

However, fear not. There are other Bookworms.

On 2 May 1898, George Bernard Shaw wrote his farewell article in the *Saturday Review*: "The younger generation is knocking at the door," he wrote, "and as I open it there steps sprightly in the incomparable Max."

I am not GBS, and so no mere Max Beerbohm will follow me. Rather, in the April issue, there steps sprightly in the incomparable Æleen Frisch.

And now to the books . . .

SECURITY

Over a year ago, I reviewed Bishop's massive *Computer Security*. The volume to hand—a mere two-thirds the size—is a revised and cut-down version. As near as I can tell, Bishop has removed much of the mathematics and some of the explanatory material; the two chapters (on assurance) by Elisabeth Sullivan have been retained.

This results in a volume that is much less a university-level textbook, and more the sort of thing that a practitioner might read.

If you have been eager to delve into computer security in a systematic way, this is the book for you. It will not grow out of date, because it deals with principles. It is chock-full of examples. Bishop has done the field a great service.

A very different tack is the one taken by Marcus Ranum. His book is an out-and-out rant concerning *The Myth of Homeland Security*. I can't say I loved every page: a number of pages actually frightened me. But Ranum does a great job. To quote him: "Researching the Department of Homeland Security, the FBI, CIA, INS, the PATRIOT Act, and so forth, one falls into a rabbit's hole of interdependent lameness and dysfunction." Yep. Ranum is neither as focused nor as self-confident as Schneier, but his book is well worth reading. I read most of my copy sitting in Dulles, having had to take off my shoes despite the fact that neither the magnetometer nor the wand had been triggered. I felt much safer.

YET MORE PENGUINS

The LSB is an innovation that has enabled tens of thousands of programmers to write in confidence, rather than insecurity. In a mere 14 chapters, the Core Team has delivered methodologies for creating, testing, and certifying code that will be LSB 2.0 compliant. I wrote about LSB 1.0 several years ago. I'm thrilled that we now have 2.0, and application programmers will be happy to see this book.

Do you read *LJ*? Do you have a shelf or a stack of old issues? Well, throw 'em out! Get the *Linux Journal Archive*, a CD-ROM containing all 116 issues from March 1994 through December 2003. I've been reading it on Mozilla. I'm told it's a snap on Lynx or Konqueror. I'm sure Firefox'll be just fine. There's even a rumor that you can use IE. And it's really useful. I reread stuff by Doc Searls and Marcel Gagne. I even reread some stuff of mine.

LOW-HANGING FRUIT

I think you can get *The Cult of Mac* even if you don't use one. Perhaps not. I'm not a member of the cult. But I have used a Mac, and I've authorized purchases of cubes for folks working for me. This book is fun even for a guy like me.

It looks like an issue of Wired. Great typefonts. Outstanding pictures. Well printed. Hardbound. On top of that, there's content. That's right. Unlike many illustrated books, this one has substance.

Get it for someone you love as a Valentine's Day present!

Ave atque vale!

book reviews

SPAM KINGS: THE REAL STORY BEHIND THE HIGH-ROLLING HUCKSTERS PUSHING PORN, PILLS, AND %*@)# ENLARGEMENTS
Brian S. McWilliams

Sebastopol, CA: O'Reilly and Associates, 2004. 256 pp. ISBN 0596007329.

Reviewed by Rik Farrow
rik@spirit.com

I had just finished burning a CD with seven months' worth of spam (740 MB), part of a project I had started last year on the sources of spam relays, when I decided to read *Spam Kings*. I wondered whether McWilliams' book would provide any insight into the minds of the people who spam. It does that, and more.

McWilliams is an investigative reporter who has appeared in the public eye before with his writing as well as his exploits, such as publishing the contents of Saddam Hussein's email. In *Spam Kings* he has created a detailed narrative that brings the lives of both spammers and anti-spammers into focus. He spins his research into the business of spamming into colorful and sometimes astonishing accounts of spammers. McWilliams' main character, David Hawke (a.k.a. Britt Greenbaum, Walter Smith, Michael Girdley, Bo Decker), is a chess-playing, ex-neo-Nazi health fanatic whose spamming exploits begin in a double-wide in South Carolina and continue on the mountain slopes of the Northeast. Hawke not only makes a good living "working" part-time, but also shares his skills with young chess players he meets at tournaments.

McWilliams also examines the ranks of spam fighters, following them as they spar online or do a better job of tracking down spammers than the state authorities who occasionally arrest and prosecute them. Being an active anti-spam activist isn't much fun: it requires lots of hard work, determination, and a thick skin to take the online insults. One advocate gets into her car only to discover a bullet hole in the windshield, and many others receive death threats by phone.

As Hawke writes in his manual for spammers, *The Bulkbook*, spamming (as well as anti-spam activities) is not for everyone: "If you are bothered by complaints or easily swayed, then you should stop reading this immediately and find another plan for making money."

Spam Kings will not teach you the technical details of spamming, but it will introduce you to the social and some of the economic details. For example, a response rate of 0.2% is considered very good. And sex, or, rather, promises that your sexual experiences will be increased or improved, sells.

McWilliams scribes a fascinating story that I found easy to read. If you have ever wanted to know more about the seamy underground of spamming, this is the book for you.

Two on Mobility

MOBILE APPLICATIONS: ARCHITECTURE, DESIGN AND DEVELOPMENT
Valentino Lee, Heather Schneider, and Robbie Schell

Upper Saddle Hill, NJ: Prentice-Hall, 2004. 331 pages. ISBN 0-13-117263-8.

MOBILE IPV6: MOBILITY IN A WIRELESS INTERNET
Hesham Soliman

Boston: Addison-Wesley, 2004. Pp. 338. ISBN 0-201-78897-7.

Reviewed by Chuck Hardin
chardin@naming-schemes.org

As the number of wireless devices connected to the Internet increases, so does the number of applications designed for such devices and the requirement to provide them with reasonable IP connectivity. It's therefore unsurprising that there are more books covering these subjects and that they display a wide range of quality.

Soliman's *Mobile IPv6* covers the subject of the IETF draft standards for mobile extensions to the IPv6 address space. These extensions are intended to provide clients with session continuity and full reachability as they go from network to network.

Unfortunately, the book does not provide a similar continuity as it goes from subject to subject, nor is its coverage of basic material complete. The summary of Internet protocol layers, for example, lists only five layers; I think the OSI would like to know where the other two layers went. The material in the chapter on security is rather disjointed. I'm not sure why the basics of public key cryptography are covered in such superficial detail, just enough to get the basic concepts but not enough to implement anything useful.

Mobile Applications is significantly better organized. It clearly achieves its purpose, which is to describe approaches to designing applications to be run on wireless devices. It does not cover actual implementation of such systems in detail, but that is not its stated purpose. Although it is ostensibly a Hewlett-Packard publication, the book does not dwell exclusively on HP's solutions in this space; there's one mention of their OpenView monitoring product, but that's it.

The striking thing about both books is their emphasis on design principles that seem good regardless of whether one is working with wireless networks or not. When *Mobile Applications* urges the reader to design with a range of browsers

in mind or to avoid requiring the use of plug-ins that may not be available to many users, or when *Mobile IPv6* recommends familiarizing oneself with IETF standards, I can only reply, "Amen." Perhaps wireless technology is temporarily forcing programmers to adopt better habits in some ways.

I hope that Soliman improves his book; it discusses a subject that deserves good coverage, and with a tighter focus and better organization, it would provide that. *Mobile Applications*, on the other hand, is a fine reference work as it is.

USENIX Membership Update

Membership renewal information, notices, and receipts are now being sent to you electronically! Remember to print your electronic receipt if you need one when you receive the confirmation email.

If you have not provided us with an email address, you are welcome to update your record online.

See **http://www.usenix.org/membership**.

You are welcome to print your membership card online as well. The online cards have a new design with updated logos—all you have to do is print!

Please note that some annual membership dues will increase as of Feb. 1, 2005:

- Individual membership dues: currently $110, will be $115.
- Educational membership dues: currently $230, will be $250.
- Corporate membership dues: currently $440, will be $460.
- Student and Supporting memberships will remain at their current rates.
- SAGE membership dues will remain the same at $40.

USENIX notes

USENIX MEMBER BENEFITS

Members of the USENIX Association receive the following benefits:

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

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

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

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

DISCOUNTS on registration fees for all USENIX conferences.

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

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

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

USENIX BOARD OF DIRECTORS

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

PRESIDENT

Michael B. Jones, *mike@usenix.org*

VICE PRESIDENT

Clem Cole, *clem@usenix.org*

SECRETARY

Alva Couch, *alva@usenix.org*

TREASURER

Theodore Ts'o, *ted@usenix.org*

DIRECTORS

Matt Blaze, *matt@usenix.org*

Jon "maddog" Hall, *maddog@usenix.org*

Geoff Halprin, *geoff@usenix.org*

Marshall Kirk McKusick, *kirk@usenix.org*

EXECUTIVE DIRECTOR

Ellie Young, *ellie@usenix.org*

20 Years Ago . . . and More

Peter H. Salus
peter@usenix.org

January 1985. 1200 USENIX members in Dallas for the Winter Conference. Rob Kolstad gave the keynote. Lauren Weinstein gave a progress report on Stargate. Susan Nycum spoke about liability issues in Netnews transmissions.

Gad! Some topics just don't go away.

Doug Comer and Ralph Droms talked about tilde trees; there was a first paper on NFS; Steve Johnson chaired an impressive languages session (DIBOL, Modula-2, Concurrent C); papers by Ian Darwin and Geoff Collyer, by Ray Essick, by Peter Honeyman and Pat Parseghian. A status report on the USENIX UUCP project by Karen Summers-Horton and Mark Horton. Really good stuff. Meaty stuff. Oh, yeah. And Mike O'Brien on mail in CSNET.

Which reminds me.

In last June's *;login:*, I celebrated the 1984 Technical Conference. It stimulated Mike O'Brien to send the following email. (Incidentally, when Mike was a graduate student, it was he who produced the first USENIX distribution tapes; Mike also was the distributor of the 50-bugs tape ["by special request"] in 1976.) Mike writes:

> I wanted to note my memories of the Salt Lake City conference. If I remember correctly, this conference was the pinnacle in the series of by-then-traditional "wizards" parties." For some years, the hosting wizards at each conference had tried to outdo previous hosts in providing an exotic, elaborate party. The tradition probably began at the Santa Monica conference, when Dave Yost hosted the party at his home in the Hollywood Hills, providing a tank of

helium, a tank of oxygen, and a box full of balloons.

At this conference, the party was held in a ski cabin owned by the university, up by Snowbird or some such. To get there was quite a drive, the last part of it through the woods on a snow-covered hillside. The cabin itself was quite rustic.

No sooner had I come in the front door than a bunch of wizards, who were all gathered near the back door, asked me, "Want something to drink?"

"Not just yet, thanks," I replied.

"No, no, you want something to drink!"

"Nope, sorry, maybe later."

"You don't understand. You really want something to drink!"

"All right, all right, I want something to drink!"

"OK! Take your pick!"

One of them swung open the back door, to reveal a snowbank up to the roofline. Holes had been punched in the snowbank, and bottles of various elixirs had been stuffed into the holes, so that the picture presented was of a glowing white wine-rack.

Things only got better from there.

(For a picture of Dennis Ritchie with Bill Joy and of Tom Ferrin with Mike Karels, taken at Snowbird, see *A Quarter Century of UNIX*.)

Summary of USENIX Board of Directors Actions

Tara Mulligan

The following actions were taken by the USENIX Board of Directors, 6/27/04–12/10/04.

REQUESTS FOR FUNDING, 2005

In 2005, $48,000 will be budgeted for standards efforts to support new developments in the Linux Standards Base, participation in POSIX IR meetings, representation at the International Standards Organization meetings, membership in The Open Group, and collaboration with the Free Standards Group.

USENIX will continue to support the USA Computing Olympiad in the amount of $15,000. USACO supports pre-college students who show great promise for careers in computer science.

The Board approved a donation of $50,000 to the Electronic Frontier Foundation in partial fulfillment of its 2001 commitment of $150,000.

USENIX will fund the Computing Research Association Committee on the Status of Women in Computing Research in the amount of $40,000. It will sponsor systems students in their Distributed Mentoring Program. For more information, see http://www.cra.org/Activities/craw/dmp/index.php.

CONFERENCES

USENIX will sponsor a new workshop, Steps to Reducing Unwanted Traffic on the Internet (SRUTI), July 7–8, 2005, concerned with finding ways to stop attacks on several standard Internet protocols: see http://www.usenix.org/events/sruti05/.

The Virtual Execution Environments (VEE) Conference in June 2005 in Chicago will merge the USENIX Virtual Machine and Technology Symposium (VM) and the ACM SIGPLAN Workshop on Interpreters, Virtual Machines, and Emulators (IVME). It was felt that researchers working on virtual execution environments would be better served by having a single top-notch event, rather than two smaller ones.

CONFERENCE REGISTRATION FEES

Registration fees for 2005 conferences were raised slightly to keep up with rising costs as follows:

Annual Tech and **LISA** Tutorials: 1/2 day, $325; 1 day, $625: 2 days, $1,200; 3 days, $1,775; 4 days, $2,300; 5 days, $2,825; 6 days (LISA only), $3,150; students, $200/day

LISA: half Technical Session/half Training day, $425

Annual Tech and **LISA** Technical Sessions, members: 1 day, $250; 2 days, $500; 3 days, $650; students, $90/day

Security Tutorials: 1 day, $650; 2 days, $1,250; students, $200/day

Security Technical Sessions: members, $675; students, $270

Technical Sessions for **all 3-day conferences with no tutorials**: members, $675; students, $270

MEMBERSHIP DUES

The Board voted to raise member dues for the first time in three years: Individual, $115 (increase of $5); Educational, $250 (increase of $20); Corporate, $460 (increase of $20); Affiliate, $110 (increase of $5). Student, Supporting, and SAGE dues are unchanged.

SAGE

The SAGE Transition Team (Geoff Halprin, Trey Harris, David Parter, and Lorette Cheswick) is pursuing setting up SAGE as an independent entity. An application for 501(c)(3) non-profit status has been filed with the IRS, and a Letter of Determination as to whether that status will be granted is pending. The USENIX office continues to provide all SAGE services and benefits.

COMMITTEES

The Board will form an audit committee in 2005, to comply with new California legislation regarding non-profit organizations.

NEXT MEETING

The next regular meeting of the Board of Directors will be held on Monday, April 11, 2005, at the USENIX Annual Technical Conference in Anaheim, CA. The Board will also meet on April 12, to discuss strategic directions.

conference reports

- Our hearty thanks go:

To the LISA '04 scribe coordinator:

John Sechrest

To the LISA '04 summarizers:

Tristan Brown

Rebecca Camus

Andrew Echols

John Hawkins

Jimmy Kaplowitz

Peter H. Salus

John Sechrest

Josh Simon

Gopalan Sivathanu

Josh Whitlock

And to the EuroBSDCon 2004 summarizer:

Jan Schaumann

LISA '04: 18th Large Installation System Administration Conference

Atlanta, Georgia

November 14–19, 2004

SPECIAL AWARDS

Doug Hughes was the recipient of the first Chuck Yerkes Award for Outstanding Individual Contribution on Member Forums, and Brent Chapman was the recipient of the SAGE Outstanding Achievement Award.

CONFERENCE SUMMARIES

- ***Configuration Management Workshop***

Paul Anderson, University of Edinburgh

Summarized by John Hawkins

This year's configuration workshop was attended by 27 people, from a range of academic and commercial organizations with often widely differing requirements for their configuration management tools.

In the three years since the first workshop, there has been great-er recognition that the configuration problem extends beyond that of configuring single machines toward methods of managing collections of nodes, often in a decentralized manner or by a devolved management team.

This workshop took a slightly different form from previous ones. Each of the four sessions followed a different theme, concentrating on separate areas of the problem. The traditional presentations of attendees' tools were not present this time, helping to avoid the "tool wars" of previous workshops. It is now widely agreed that no current tool adequately solves all the problems in configuration management, and that collaboration between researchers in the field and a refocusing on the principles behind tool design, rather than the refinement of any one tool, are required for future progress.

As usual, a range of polls were taken. The majority of attendees regard themselves as tool developers, but a much smaller number have written tools that are also used by others. Many manage either high-performance clusters or Windows machines, and well over half indicated a strong interest in the theoretical research issues in system configuration, in addition to practical concerns of tool development.

Some attendees felt that those within the configuration community are approaching a consensus, but others thought there is some way to go yet. The problems involved are only beginning to be specified in words whose meanings are agreed on, and there is still much duplication of work.

Attempts were made to define a number of terms in common usage but with slightly ambiguous or overloaded meanings. "Policy" was suggested to be a description of what a machine is intended to do, including intended collaborations as well as configurations. "Service Level Agreements" (SLAs) were defined as relationships promising service within specific tolerances.

Effort is required on the specification of intermachine relationships. A tool may correctly configure a machine in isolation, but more complex ways of capturing the details of a service specification are needed to ensure that intermachine relationships hold, thus producing the desired service behaviors.

SLAs cross a dividing line, since they require active monitoring. While it is essential that this monitoring be integrated into the tool, it should be part of a layer distinct from the specification language currently used.

As soon as dynamic properties are introduced, much of the certainty that previously existed is lost, and it will no longer be possible to tell what's true or false at any particular time, thus moving into the realm of probabilities. This is a fundamental problem that must be lived with. It was suggested that this uncertainty has at least two dimensions, time and value uncertainty.

The possibility of a set of standards for configuration was discussed, with the POSIX standard for UNIX as an analogy. The use of a low-level API for configuration was also suggested.

Mark Burgess led a session on decentralized configuration, illustrated by Ed Smith's simulation of a decentralized service manager capable of reconfiguring in response to node failure.

A move away from centrally managed systems is required to cope both with problems of scale and with vulnerability to central points of failure that are becoming problematic for large sites. However, this move brings with it reductions in predictability, trust, and relative simplicity of management. Decentralized management allows autonomy where some configuration decisions are made by the system by use of protocols, including those for negotiation and service discovery. To govern this autonomous behavior, the system must have in place an awareness of its environment that is unnecessary under central management and policy control.

There was some discussion of pervasive computing, the management of large numbers of small devices, but this is currently an open problem and it may be too early to consider it in any detail since the range of requirements of such systems is not yet fully understood.

Luke Kanies and Alva Couch talked of the difficulty of achieving more widespread adoption of configuration management tools. Barriers to adoption include the hostility of system administrators used to the current ways of working, the complex task of respecifying the configuration of the site under the new tool, and unhelpful management attitudes not aided by poor cost models and lack of trust in often inadequately proven systems.

Alva's presentation described how the cost of configuration management goes through four phases, each phase reaching a point where the cost rapidly increases and a more sophisticated approach to configuration management must be adopted. Most sites have reached the point at the end of the second phase where the configuration is managed "incrementally," for example with cfengine, but encounters problems with hidden preconditions requiring bare-metal rebuilds. They are not aware of how to make the transition to a "proscriptive" management strategy.

Site managers need to know at what point it becomes more economical to adopt a heavyweight tool such as LCFG, with huge initial costs in setup and staff training but more robust results in handling large numbers of machines.

The problem of loss of institutional memory between the "incremental" and "proscriptive" phases would be alleviated if data mining techniques could be used to pull out a large proportion of the current configuration and convert this to configuration data for the new system. There was some discussion as to whether this is intractable.

A session on case studies with Steven Jenkins and John Hawkins attempted to increase the number of specific examples of configuration problems tool users are actually grappling with. Questionnaires were distributed, and although these have yet to be analyzed at the time of writing, the response was impressive, and it is hoped that this exercise will prove fruitful.

On devolved aspect resolution, it was suggested that the system should follow the human process of resolution. Political lines are important and should be reflected by the machine aspect resolution. An expert system could be utilized to predict the impact of the choice of value.

Suggestions about where research should focus from this point included the formalization and documentation of the collective knowledge so far, provision of limited user control, description of conflicts within configurations and procedures for their resolution, mechanisms for configuration transactions, and the identification of a wider range of case-study examples with a variety of candidate tools on which to try them.

This is likely to remain an active area for some time, but there was a general feeling of optimism at the workshop that solutions to many of the problems are reachable.

- ***Sysadmin Education Workshop***

  John Sechrest, PEAK Internet Services; Curt Freeland, University of Notre Dame

  Summarized by John Sechrest

  The system administration education workshop addresses the process of system administration education at the university level. This was the seventh year of the workshop. Previous materials for system administration course content were discussed and an earlier curriculum of how many different courses might fit together into a degree program was reviewed.

A new Web site (http://education.sage.org) was unveiled as a starting point for information collection, and an online content management tool called Drupal was explored (http://education.sage.org/drupal).

The goal of the Web site is to enable more collaboration and cooperation between groups working on university sysadmin education.

Over the last two years, there has been a reduction in the number of universities that support system administration courses. The changes in the economy are being felt in the universities.

There was a substantial discussion of just-in-time learning materials for university training for existing system administration staff and how these materials might be shared in the context of other courses. There was also discussion of how online learning modules might be useful.

The system administration education mailing list is now at sysadm-education@sage.org.

- ***Advanced Topics Workshop***

Adam S. Moskowitz, Menlo Computing

Summarized by Josh Simon

The Advanced Topics Workshop started with a quick overview of the new moderation software and continued with introductions around the room—businesses (including consultants) outnumbered universities by about 2 to 1, and the room included three former LISA program chairs and six former members of the USENIX Board or SAGE Executive Committee.

Our first topic was introducing the concepts of disciplined infrastructure to people (i.e., it's more than just cfengine or isconf, or infrastructure advocacy, or getting rid of the ad hoc aspects). Some environments have solved this problem at varying levels of scale; others have the fear of change paralyzing the system administration staff. One idea is to offload the "easy" tasks either to automation (while avoiding the "one-off" problem and being careful with naming standards) or to more junior staff so that senior staff can spend their time on more interesting things. Management buy-in is essential; exposing all concerned to LISA papers and books in the field has helped in some environments. Like many of our problems, this is a sociological one and not just a technical one. Remember that what works on systems (e.g., UNIX and Windows boxes) may not work for networks (e.g., routers and switches), which may be a challenge for some of us. We also noted that understanding infrastructures and scalability is very important, regardless of whether you're in systems, network, or development. Similarly important is remembering two things: First, ego is not relevant, code isn't perfect, and a developer's ego does not belong in the code. Second, the perfect is the enemy of the good; sometimes you have to acknowledge there are bugs and release it anyway.

After the morning break, we discussed self-service, where sysadmin tasks are traditionally handed off (ideally in a secure manner) to users. Ignoring for the moment special considerations (like HIPAA and SOX), what can we do about self-service? A lot of folks are using a number of Web forms or automated emails, including the business process (e.g., approvals), not just the request itself. One concern is to make sure the process is well defined (all edge cases and contingencies planned for). We've also got people doing user education ("we've done the work, but if you want to do it yourself the command is…"). Constraining possibilities to do only the right thing, not the wrong thing, is a big win here.

Next we discussed metrics. Some managers believe you have to measure something before you can control it. What does this mean? Well, there are metrics for service goals (availability and reliability are the big two), in-person meetings for when levels aren't met, and so on. Do the metrics help the SAs at all, or just management? It can help the SAs identify a flaw in procedures or infrastructure, or show an area for improvement (such as new hardware purchases or upgrades). We want to stress that you can't measure what you can't describe. Do any metrics other than "customer satisfaction" really matter? Measure what people want to know about or are complaining about; don't just measure everything and try to figure out what's wrong from the (reams of) data. Also, measuring how quickly a ticket got closed is meaningless: Was the problem resolved, or was the ticket closed? Was the ticket reopened? Was it reopened because of a failure in work we did, or because the user had a similar problem and didn't open a new ticket? What's the purpose of the metrics? Are we adding people or laying them off? Quantifying behavior of systems is easy; quantifying behavior of people (which is the real problem here) is very hard. But tailor the result in the language of the audience, not just numbers. Most metrics that are managed and monitored centrally have no meaningful value; metrics cannot stand alone, but need context to be meaningful. Some problems have technical solutions, but metrics is not one of them. What about trending? How often and how long do you have to measure something before it becomes relevant? Not all metrics are immediate.

After a little bit of network troubleshooting (someone's Windows XP box was probing port 445 on every IP address in the network from the ATW), we next discussed virtualized commodities such as User Mode Linux. Virtual machines have their uses—for

research, for subdividing machines, for providing easily wiped generic systems for firewalls or DMZ'd servers where you worry about them being hacked, and so on. There are still risks, though, with reliance on a single point of failure (the hardware machine) theoretically impacting multiple services on multiple (virtual) machines.

Next we discussed how to get the most out of wikis as internal tools. What's out there better than TWiki? We want authentication out of LDAP/AD/Kerberos, among other things. The conference used PurpleWiki, which seems to be more usable. There's a lot of push-back until there's familiarity. They're designed for some specific things, but not everything. You need to be able to pause and refactor discussions if you use it as, for example, an email-to-Wiki gateway. (There is an email-to-Wiki gateway that Sechrest wrote.) If email is the tool most people use, merging email into a wiki may be a big win. Leading by example—take notes in the wiki in real time, format after the fact, organize it after you're done—may help sell it to your coworkers.

Next we listed our favorite tool of the past year, as well as shorter discussions about Solaris 10, IPv6, laptop vendors, backups, and what's likely to affect us on the technology front next year. We finished off by making our annual predictions and reviewing last year's predictions; we generally did pretty well.

KEYNOTE ADDRESS

Going Digital at CNN

Howard Ginsberg, CNN

Summarized by Gopalan Sivathanu

Howard spoke about CNN's plan to move video storage, playout, and editing from physical media to file-based operations on disk. The main motivating factors behind this change are improving the speed of general operations like editing and transfer and bringing about a better means for locating archived data. He pointed out that through file operations, searching becomes much easier than pulling out physical videotapes.

According to CNN, using digital format for storing video simplifies access, eases manageability, and provides ready adaptability and a competitive edge in expediting the operations so as to bring news at the right time. Howard introduced the method that CNN plans to adopt to bring about this change: a system called "Integrated Production Environment" (IPE), whose operations would broadly be managing media assets, scheduling the various operations on the media, production, playout, and, finally, archiving the old data so that they can be accessed quickly whenever required.

While speaking about the complexity involved in such a huge transition, Howard presented numbers for the approximate storage space required for storing one day's video in high-res and low-res formats. He pointed out that, since high-res video might require around 22GB of storage per day, performing operations like editing and file transfer required for acquiring, archiving, and transferring data is a big challenge.

Howard then discussed the various pros and cons of making this transition. The major pros he pointed out are parallelism in recording and editing, sharing, versioning, flexibility, and transfer speed. The main disadvantages are absence of "at scale" reference architectures and the asynchronicity between deployment and technological improvement wherein technological development renders the system obsolete by the time it's deployed.

Howard pointed out that providing forward and backward compatibility between software and hardware is one of the important challenges in making the transition. He then described the project phases and workflow and gave a brief overview of the system architecture.

SPAM/EMAIL

Scalable Centralized Bayesian Spam Mitigation with Bogofilter

Awarded Best Paper!

Jeremy Blosser and David Josephsen, VHA Inc.

Summarized by Gopalan Sivathanu

Jeremy and David presented the paper together. Jeremy began by describing the spam problem and existing solutions such as checksumming and sender verification. Before introducing their own solution, he went into the history of Bayesian filtering and discussed its disadvantages. He then described their Bogofilter Bayesian classifier, which they implemented on a Linux and Qmail plaform. He gave the salient features of the method and its capabilities. They believe the success rate to be as high as 98–99%.

They displayed a graph showing the percentage of inbound mail blocked as spam to total inbound mail before and after implementing Bogofilter. It showed a sharp increase in the number of mails filtered after Bogofilter was installed, in late April of 2003, and the performance has persisted for over a year.

Jeremy and David described the training phase required for Bogofilter: sorting several days' worth of messages sorted into spam and non-spam teaches Bogofilter how to differentiate spam in a given environment. They explained that, after training, the parameters of Bogofilter have to be tuned by running it over a sample of presorted mail and log output and comparing error rates. They gave a 10-minute demonstra-

tion of their automated training scripts.

INVITED TALKS

- ***What Is This Thing Called System Configuration?***

  Alva Couch, Tufts University

  Summarized by Jimmy Kaplowitz

  Alva Couch talked about efforts to find a "good enough" path to configuration management. What would be ideal would be to describe to the computer the desired high-level behavior, but nowadays we still manually translate this into an implementation specification for the computers to interpret. Couch distinguished between host-based configuration, where individual machines are set up to cooperate in providing a service, and the more fault-tolerant network-based method of configuration, where a service is set up network-wide. The languages used in system configuration can either be procedural, an intuitive paradigm that shows the steps of implementation that lead to the desired result, or declarative, a nonobvious but clear form of language that simply states the desired intent and leaves it to software to determine the implementation. These contrasts reflect a continuum of rigor leading from manual and unstructured changes on individual hosts to declarative specifications applied to the entire fabric of the network. The most important aspect of a configuration system, however, is the discipline of the administrators adopting it. This is much more important than the software involved.

  Configuration specifications can either be proscriptive, specifying everything, or incremental, where some requirements are specified but others are not. Beginners, Couch says, are not proscriptive enough, allowing the presence of latent preconditions that differ among hosts. The next step beyond this is a federated network, where software can change the function of individual machines in response to usage patterns and other conditions. As network management systems gain complexity from ad hoc to federated, they become harder to start managing but easier to continue and finish managing. Simpler configuration management systems are better on smaller-scale networks and over shorter periods of time, but in large and long-lived networks these techniques help.

  Configuration languages, at their core, describe data and state, not the algorithms with which programming languages concern themselves. Existing languages to manage federated networks have classes of machines, but they are insufficient when the classes overlap. The best solutions use aspect composition to satisfy constraints that specify what properties the administrators require the network configuration to possess. These constraint-based languages are useful but hard. Configuration language theory is still in its infancy.

- ***Anomaly Detection: Whatever Happened to Computer Immunology?***

  Mark Burgess, Oslo University College

  Summarized by Jimmy Kaplowitz

  Anomaly detection in the context of computers is the process of monitoring systems and looking for anything unusual, or "funny," and maybe fixing it, too. To do this it is necessary to determine what an anomaly is. Are intrusions anomalies? What about viruses and malware? Should the system signal that regulation is needed? Regardless, the management of the anomaly detection system should be simple and flexible.

  Burgess emphasized the scientific approach to this problem. He says there are two types of knowledge: uncertain knowledge about the real world, such as that found in biology and experimental physics, and certain knowledge about the fantasy worlds of math and modeling. It is hard to define what a pattern is that an anomaly detection system would be looking for, but a rough guess would be a noticeable repetition or continuous variation. This is modeled with rules, and the knowledge we have about normal behavior is modeled with parameterized expressions.

  Anomalies can be modeled as discrete events. For this task, one can use languages anywhere on the Chomsky hierarchy of languages, ranging from regular languages to recursively enumerable languages. In practice, all discrete anomalies are finite in length, so regular languages and the regular expressions that comprise them will suffice. It is also possible to use techniques such as epitope matching or probabilistic searching of the pattern space.

  Continuous modeling of anomalies is a high-level way to attack the problem. This always involves intrinsic uncertainty, since each point represents a summary of many values. It is possible to include time in the model, considering sequences rather than individual anomalous symbols, but remembering all things doesn't usually help. On the other hand, forgetting temporal details implies certainty.

  In fact, whether the model is discrete or continuous, there will always be uncertainty, either over whether the symbol is correctly matched or over whether the shape of the trend is significant. How to deal with this uncertainty is an important matter of policy. One example of a policy question is where to set thresholds. Small variations should be ignored, but striking differences need to be dealt with. Also, one can focus on high-entropy anomalies, which are very broad, or specific and focused low-entropy anomalies.

Burgess' conclusion re-emphasizes that there is inevitable uncertainty in anomaly detection and that policy is necessary to resolve it. We lack causally motivated models to explain the reasons behind the anomalies that occur. Human beings are still the state of the art for anomaly detection. It is important not to make the sample size too large when determining whether an event is anomalous or not, since with a large enough sample nearly everything looks like normal variation. Burgess finished by saying that the next step for anomaly detection is to develop higher-level languages to describe policy.

- ***What Information Security Laws Mean for You***

John Nicholson, Shaw Pittman

Summarized by Rebecca Camus

With the prevalence of e-commerce over the past several years, more and more emphasis has been placed on guaranteeing consumers that their personal information will be secure in the business's databases and while the customer is performing online transactions. John Nicholson addressed the topic of information security by first giving the user a crash course in civics, then presenting both federal and state-level regulations and laws that businesses must comply with, and finally discussing what the government is doing to enforce these regulations.

Nicholson began by delineating the differences between state and federal regulations and how federal and state laws interact. Federal law has the power to preempt state law on controversial issues relating to individual protections. However, states have the power to make these laws stricter if they wish.

The next topic that Nicholson covered regarded federal laws. In the past few years, the federal government has passed several information security laws that all businesses must comply with. The ones presented in this lecture and brief explanations of each are:

Federal Information Security Management Act: requires each business to inventory their computer systems, identify and provide appropriate security protections, and develop, document, and implement information security programs.

Gramm-Leach-Bliley Act: requires financial institutions to protect nonpublic financial information by developing, implementing, and maintaining their information security programs.

Health Insurance Portability and Accountancy Act: requires all health-related industries (including any company that deals with any health-related businesses or companies that self-insure their employees) to install safeguards in protecting the security and confidentiality of their customers' identifiable information.

Sarbanes-Oxley Act: requires that all stored information must be accurate. In addition to the requirements of each law, they all require that companies perform frequent testing and risk assessments and fix all subsequent problems.

Prompted by the increase in identity thefts, California has led the way with state-level information security laws. Even businesses not located in California but with customers who are California residents must comply with California's new information security laws. According to California's SB 1386, a resident cannot have his or her full name or first initial and last name connected with either his or her Social Security Number, driver's license number, California identification card number, or account/
credit card/debit card number, access code, or password. In addition to SB 1386, California has also passed AB 1950, which requires all businesses (even third-party companies who have purchased information about California residents) dealing with California residents to implement and maintain reasonable security procedures to protect the information from unauthorized access or modification.

All of these regulations are being heavily monitored and enforced by the Federal Trade Commission. Businesses that violate any of these laws or regulations are subject to heavy fines and possible lawsuits.

INTRUSION AND VULNERABILITY DETECTION

Summarized by Josh Whitlock

- ***A Machine-Oriented Vulnerability Database for Automated Vulnerability Detection and Processing***

Sufatrio, Temasek Laboratories, National University of Singapore; Roland H. C. Yap, School of Computing, National University of Singapore; Liming Zhong, Quantiq International

The motivation for the work is that CERT-reported vulnerabilities have increased greatly in the period from 1995 to 2002. The increase, at first glance, is exponential; there have been decreased reports in the time period, though. To address this increase in vulnerabilities, system administrators look at vulnerability reports and use databases of vulnerabilities. However, such human intervention is not scalable to the increase in the number of vulnerabilities. The solution is to increase the speed of addressing new alerts by moving from human language to machine language reports. The goal of the work is to create a new framework and database that can be used by third-party tools by making the framework declarative and flexible.

Related work on this subject includes Purdue Coop Vulnerability database, ICAT, Windows Update, and Windows Software Update Server. ICAT is not designed for automatic responses

but is geared more toward mere vulnerability searches. Windows Update and Windows Software Update Server, two automatic patch update systems, are closed and have black-box updates. Issues with these products mentioned above include concerns that black-box scanners and updaters might leak information and do not address problems such as what is affected by the vulnerability in a system.

To produce a prototype of the framework, approximately 900 CERT advisories were examined manually. Information scavenged from the advisories included conditions for vulnerabilities to exist and the impact of vulnerabilities. A scanner written in Perl and a mySQL database were used as the prototype. A symbolic language was created and used to store a vulnerability source, an environment source, a vulnerability consequence, and an exploit.

The conclusions reached were that manual translations of vulnerabilities were not feasible due to the increasing size; the key to a better framework is the abstraction of vulnerabilities instead of new tools, and this issue needs the involvement of many organizations, including CERT, BugTraq, and SANS.

- ***DigSig: Runtime Authentification of Binaries at Kernel Level***

  Axelle Apvrille, Trusted Logic; Serge Hallyn, IBM LTC; David Gordon, Makan Pourzand, and Vincent Roy, Ericsson

  DigSig is a kernel module whose main intent is to provide protection from trojan horses. The module achieves this goal using public key cryptography, signing known trusted programs using a hidden private key, and verifying signatures at program load using the public key. Previous attempts include CryptoMark and modules from IBM Research, but DigSig seeks to contribute by being practical, simple to install, and efficient.

  DigSig uses SHA-1 and RSA encryption, supports signature revocation, and has good performance. The main problems for DigSig include the lack of support for network file systems, lack of script handling (allowing for trojaned scripts going uncaught), and no protection against vulnerabilities such as buffer overflows.

- ***I³FS: An In-Kernel Integrity Checker and Intrusion Detection File System***

  Swapnil Patil, Anand Kashyap, Gopalan Sivathanu, and Erez Zadok, Stony Brook University

  The motivation for I³FS (pronounced I cubed FS) is that intrusions are on the rise; prevention is nearly impossible, and effective intrusion handling requires timely detection. The most common ways to detect intrusions are to establish invariant, run-scheduled integrity checks and to use non-kernel programs for the detection. I³FS does all this by existing in the kernel at the file system layer and using checksums for integrity checks. A look at other tools shows that many are user tools, including Tripwire, Samhain, and AIDE. The Linux Intrusion Detection System is one of the few kernel tools. I³FS was designed with a threat model of unauthorized replacement of key files, modification of files through raw disk access, and modification of data in the network. I³FS has a stackable file system. The administrator can choose which files to protect and the policy for protecting them. The policies range from checking inode fields to the frequency at which checks are performed.

  I³FS is implemented using an in-kernel Berkeley database and a B+ tree format. With caching, performance is good.

INVITED TALKS

- ***LiveJournal's Backend and memcached: Past, Present, and Future***

  Lisa Phillips and Brad Fitzpatrick, LiveJournal.com

  Summarized by Andrew Echols

  LiveJournal started out as a college hobby project, providing blogging, forums, social networking, and aggregator services. Today, LiveJournal boasts over 5 million user accounts and 50 million page views every day. LiveJournal quickly outgrew its capacity several times over, starting with a single shared server and scaling up to five dedicated servers.

  Each step along this path had problems with reliability and points of failure, with each upgrade merely trying to remedy the problem at hand. Eventually, the user database was partitioned into separate user clusters, but as growth continued, this system became chaotic as well.

  Today, there is a global database cluster with 10 individual user database clusters, as well as separate groups of Web servers, proxy servers, and load balancers. Furthermore, master-master clusters were implemented so that each database master has an identical cold spare ready in the event of a hardware failure.

  LiveJournal also uses a large amount of custom software that has been open sourced. Perlbal is a load balancer written in Perl that provides single-threaded, event-based load balancing. Perlbal also has multiple queues for prioritizing free and paid users.

  MobileFS is a distributed user-space file system where files belong to classes. The file system tracks what devices files are on and utilizes multiple tracker databases.

  Memcached provides a distributed memory cache, taking advantage of unused system memory to reduce database hits. Memcached

is now in use at many sites, including Slashdot, Wikipedia, and Meetup.

- ***NFS, Its Applications and Future***

  Brian Pawlowski, Network Appliance

  Summarized by John Sechrest

  Brian Pawlowski is active in the development of NFS version 4. NFS is a distributed file system protocol started in 1985. The current version is version 3; the new revision of NFS, version 4, was influenced by AFS. Brian outlined how NFS was used in the past and contrasted it with how it is being used today. Grid computing and the larger Internet are putting more demands on distributed file systems.

  NFSv4, an openly specified distributed file system with reduced latency and strong security, is well suited for complex WAN deployment and firewalled architectures. NFSv4 represents a huge improvement in execution and coordination over NFSv3, also improves multi-platform support, and is extensible. It lays the groundwork for migration/replication and global naming.

  NFSv4 adds a Kerberos V5 (RFC1510) authentication system as a way to create a secure distributed file system. It also provides finer grained control for access, including optionally supporting ACLs.

  NFSv4 is available now in some platforms. More will be coming out over the next six months. It is available for Linux 2.6, but not by default; you must add it explicitly.

  Future development in NFS may include session-based NFS, directory delegations, migration/replication completion, failover, proxy NFS, and a uniform global namespace.

  Brian spent some time outlining how NFS works in a grid or clustered environment.

  NFSv4 seems like a substantial transformation, which will make a significant difference for distributed file systems.

CONFIGURATION MANAGEMENT

Summarized by Josh Whitlock

- ***Nix: A Safe and Policy-Free System for Software Deployment***

  Eelco Dolstra, Merijn de Jonge, and Eelco Visser, Utrecht University

  Transferring software from machine to machine is hard, especially when packages don't work right. There can be difficulties with multiple versions and unreliable dependency information. The central idea of Nix is to store all packages in isolation from one another so that dependency and version problems evaporate. This is done by storing the path of each package as an MD5 hash of all the inputs used to build the package, thus producing automatic versioning. When a Nix package is installed, if there are dependencies, those packages are automatically installed too. Versioning is dealt with by having symlinks that point to the current environment. To roll back to a previous version means changing a symlink to point to the old version. To delete previous versions, the symlink to the old version is removed and the version is removed from disk. Nix thus allows for safe coexistence of versions, reliable dependencies, atomic upgrades and rollbacks, and multiple concurrent configurations.

INVITED TALK

- ***Documentation***

  Mike Ciavarella, University of Melbourne

  Summarized by Rebecca Camus

  Mike Ciavarella presented on the importance of documentation to system administrators. However, instead of simply stating how essential documentation is to system administrators, he sparked many people's interests by comparing the work of a system administrator to Alice from the book *Through the Looking Glass*.

  He began the discussion by reinforcing the fact that system administrators are very different from the users they support. Many times these differences lead to a lack of communication between the user and the system administrator, which, in turn, causes the sysadmin to feel underappreciated and leads to a sense of frustration. Such sysadmins often feel little motivation to use valuable time documenting their system.

  It is essential for system administrators always to document their work, for multiple reasons. A system administrator will often be working on several projects at once, and it is very easy to forget what one has previously done on a project when dealing with several other concurrent projects. Also, the system administrator has to think of the future. It is common for things to go wrong in a system that has not been worked on recently. It is helpful to be able to reference documentation that will help solve the problem at hand. And a system administrator may move on to other projects. It is helpful to those dealing with the system to be able to reference the material that was written when the system was created.

  Overall, documentation will improve the way system administrators are perceived, emphasizing their professionalism and ultimately saving time and reducing stress.

GURU SESSION

- ***Linux***

  Bdale Garbee, HP Linux, CTO/Debian

  Summarized by Tristan Brown

  Topics included HP's commitment to Linux, stabilization of the Linux kernel, and future directions for Linux.

  HP has a huge commitment to Linux across their product line.

Every division uses Linux to some degree, although some (such as the Laptop division) would like to see more. Linux represents a source of future growth, given its current immature state in the market. One problem currently faced is the variety of distributions that are used and supported, even at HP. Not only are tier-A distributions used (e.g., SuSE or RedHat), but so are near-tier-A distributions, such as some international distributions. For many people, a noncommercial distribution like Debian is required. Efforts to create one Linux strategy are underway at HP.

The discrepancies between versions of the Linux kernel can be a problem, as many libraries and applications need to be compiled against a specific kernel. This has traditionally been a problem for Linux, although recent efforts to stabilize major interfaces have resulted in less churn. This is a good thing, due to the tendency for closed-source drivers to be compiled for specific versions. Certification is also easier, since certification is typically done for a specific kernel. The end result is that the kernel can now be considered a serious tool for end users.

There are many niches Linux is now or will be entering, beyond the typical desktop or server setup. A significant portion of high-performance computing is done using Linux clusters, and Linux is beginning to enter the financial markets. At the other end of the spectrum, HP is beginning to work with Linux in the embedded space, although not much for real-time uses. They are also taking Linux to the DSP market.

NETWORKING

- ***autoMAC: A Tool for Automating Network Moves, Adds, and Changes***

  Christopher J. Tengi, Joseph R. Crouthamel, Chris M. Miller, and Christopher M. Sanchez, Princeton University; James M. Roberts, Tufts University

  Summarized by Tristan Brown

  Network administration at Princeton's Computer Science department, with more than 1500 hosts and 100 subnets, is less than trivial. To manage this system, the autoMAC tools were developed. The toolset replaces a largely manual process of entering host information into a database, configuring a switch, and connecting a patch cable with a fully automatic approach. Key to this system are several tools:

  - Web registration system. This allows administrators and users to enter configuration information that is validated and automatically entered into the DHCP, DNS, and NIS databases. This replaces a previous manual system that involved editing a file using vi.
  - NetReg server. This server answers DHCP requests for all hosts and acts as a gateway for unauthenticated users. When an unrecognized host is attached to the network, all HTTP requests are intercepted and redirected to a registration page.
  - FreeRADIUS. This technology, originally created for wireless access points, allows a device's MAC address to be used as an authentication key without any special configuration from the client end. When the switch sees a device, it asks the server for information about its MAC address. Known devices are automatically switched to their appropriate VLAN, while unknown devices end up on the registration VLAN.

  With these tools, adding a host to the database is simple to perform, and moving from one Ethernet port to another requires no configuration changes at all. Future improvements include automatic virus scanning that moves infected hosts to a quarantine VLAN. More information is available at http://www.cs.princeton.edu/autoMAC/.

- ***More Netflow Tools for Performance and Security***

  Carrie Gates, Michael Collins, Michael Duggan, Andrew Kompanek, and Mark Thomas, Carnegie Mellon University

  Network analysis for large ISP networks can involve massive data sets consisting of over 100 million flow records per day stored for a period of months. To deal with this demand for storage and processing, the SiLK tools were developed. Based on Netflow logs, the custom SiLK format takes less than half the space and has several tools to perform statistical analysis on the data.

  The SiLK logfile format starts with the directory tree, with data segregated by log date and type of traffic. Several fields from the Netflow log are removed, and others, such as time, are stored with reduced range or precision. HTTP traffic is isolated from the rest, allowing the transport type and port number fields to be reduced to two bits of data, specifying only the TCP port used. This results in a two-byte savings over other packet types and reduces each entry to less than half the size of the original Netflow record.

  To complement the compact logfile format, SiLK includes four tools to analyze traffic patterns and seven utilities for summarization. The tools all operate across the directory structure and can work with sets of IP addresses. Analysis performed with the tool set can be used by administrators in a variety of ways, such as detecting virus traffic on the network.

  The SiLK tool set is available at http://silktools.sourceforge.net.

INVITED TALK

Flying Linux

Dan Klein, USENIX

Summarized by Tristan Brown

This talk poses a simple question: Is Linux robust enough to power the computers responsible for fly-by-wire in state-of-the-art aircraft? These computers operate with real-time constraints and no tolerance for unexpected failure. Linux faces several difficulties meeting these demands.

Linux developers work on activities they enjoy, not necessarily the drudge-work required to eliminate elusive, rarely encountered bugs. Corporate developers are paid to do the work the Linux developers don't do.

The Linux source contains millions of lines of code, many of them experimental, untested, or irrelevant to a fly-by-wire system. Knowing which components to compile in is a difficult task, and tracking the interrelations between these systems is all but impossible. A GraphViz graph of the FreeBSD kernel (fewer than half the lines of code of Linux) results in a graph so complex that individual nodes and edges are no longer distinguishable.

Linux is the most exploited operating system, counting only manual attacks. The kernel receives hundreds of patches from all over the world. Which contributors can you trust? The NSA has decided to harden Linux, but are we sure vulnerabilities haven't slipped in through a back door?

The loose, open development model of Linux isn't ideal for creating a secure, stable platform. Ignoring this, however, there is yet another problem. Fly-by-wire systems are computers: they can react only to the situations they have been programmed for. Mechanical or sensor failures can result in the computer taking the wrong course of action. Human pilots have the ability to adapt to new situations much better than their electronic counterparts, although this advantage is steadily shrinking.

The unfortunate conclusion is that Linux, like its penguin mascot, may never fly. The difficult task of creating a hardened, real-time system can be left to purpose-built software, and Linux can remain the general-purpose operating system it was designed to be.

SPAM MINI-SYMPOSIUM

Filtering, Stamping, Blocking, Anti-Spoofing: How to Stop the Spam

Joshua Goodman, Microsoft Research

Summarized by John Hawkins

This talk gave a broad picture of the difficulties currently being caused by widespread proliferation of spam, the techniques recently adopted by spammers to get around ever more complex spam-filtering tools, and possible solutions to further deal with spam.

A number of statistics were presented, including the results of a poll in which 40% of respondents stated that spam overload was the biggest problem faced by IT staff. Spam has increased from around 8% of all email in 2001 to at least 50% currently. Wasted time dealing with unwanted messages, offense caused by often obscene content, and a lowering of trust in email systems make spam a major problem.

Examples of different spammer techniques were given, such as enclosing content in an image, swapping letters of words, and using coded HTML to confuse filters. Techniques for gaining access to computers to turn them into spam relays were also mentioned.

A range of methods to deal with spam were discussed, some of which may be combined to increase effectiveness: Better filtering and use of machine-learning techniques will identify more complex patterns: using honeypots, which should never receive "good" mail, to identify messages that are definitely spam; charging small amounts per message as an economic barrier to profitable spam; computational challenges requiring a calculation on the client, making it difficult to send many messages simultaneously. Many of the techniques discussed are only applicable in certain situations, and most have significant drawbacks. Some proved unpopular with the audience, which was composed mainly of email admins.

Approaches for anti-spoofing email addresses, types of non-email spam, and legal approaches to tackling spam, such as the Can-Spam Act, were also mentioned.

The speaker pressed the case for a specific conference to cover spam-related issues, due to the increasing seriousness and complexity of the problem.

Lessons Learned Reimplementing an ISP Mail Service Infrastructure to Cope with Spam

Doug Hughes, Global Crossing

Summarized by John Hawkins

In contrast to the previous talk by Joshua Goodman, which gave a wide-angle view of current spam issues, this talk discussed the specifics of dealing with spam while managing an engineering mail platform with 200 users, a number of ISPs including one with 1096 users, and 4 to 6 million mails processed a day.

The talk began with some statistics, identifying the source and nature of the spam dealt with and how some of the trends are changing. Of the mail received from Italy, for example, 90% is currently being blocked as spam. It was found that 98% of spam could be blocked by examining the headers alone.

The main part of the talk was more technical, covering how the sites' mail systems were configured to deal with spam. The majority of the filtering employs a modified

version of smtpd using a binary search tree to store logs, which vastly reduces search time. Sender addresses can be compared with those previously seen to assess whether a message is likely to be genuine.

The system was shown to be very effective, blocking up to 98% of spam while giving virtually no false positives.

INVITED TALK

- ***Grid Computing: Just What Is It and Why Should I Care?***

Esther Filderman and Ken McInnis, Pittsburgh Supercomputing Center

Summarized by Josh Whitlock

There are different types of grid computing, including utility computing (where processing cycles are for sale), distributed computing, and high-performance resource sharing. What grid computing actually is lies between these types. Examples of grids include the CERN Large Hadron Collider Computing Grid, supporting approximately 12 petabytes of data per year with 6000+ users, and the Electronic Arts Grid for the Sims Online game, with approximately 250,000 players. The challenges of using grid computing include flexible virtual organizations having different site policies and disparate computing needs.

Components of grids include security, toolkits, job management, data movement, and Web portals. Security for grid computing is basic X.509. The standard toolkit is Globus. Condor is a job manager for grids, and while flexible and portable, it is not open source or simple to use. Data movement is supported with programs such as GridFTP, which is built on top of regular FTP but is used for high-performance, secure, and reliable data transfer. Web portals are used to provide a consistent front to a grid that makes maintenance less difficult.

Grids are most successful when they have a purpose: setting up a grid at a university that everyone can use for their own purpose is not a good idea, but setting one up for a research project is a good idea. Political challenges in running a grid include coordinating work between sites that don't trust one another and implementing common policies. Technical challenges include having interoperable software between sites and synchronizing upgrades to the grid software.

MONITORING AND TROUBLESHOOTING

Summarized by Andrew Echols

- ***FDR: A Flight Data Recorder Using Black-Box Analysis of Persistent State Changes for Managing Change and Configuration***

Chad Verbowski, John Dunagan, Brad Daniels, and Yi-Min Wang, Microsoft Research

The Flight Data Recorder concept presented in this talk targets auditing, configuration transaction, and "what if" scenarios. In auditing scenarios, the FDR would assist in troubleshooting by identifying what is on a machine, and by answering questions like what is installed, where, when, and by whom. Changes are then grouped logically into change actions.

These change actions can be treated as transactions, allowing groups of changes to be automatically rolled back if they are no longer wanted, poorly done, had an adverse effect, or were malicious.

The information gathered also aids in answering "what if" questions regarding the impact of certain changes. This requires the formation of "application manifests," which are records of an application's frequency of interactions, all states read, all states written, and how often it is run.

The approach taken in the research is to get into the lowest level of the OS. There, state interactions can be monitored and understood. To be useful, information gathered must then be analyzed and presented in a way useful to humans. This is implemented as a service that logs gathered information locally. Periodically, the logs are uploaded to central storage, where they can be processed and inserted into a database. Finally, a client may view this information online or retrieve it for offline use.

- ***Real-Time Log File Analysis Using the Simple Event Correlator (SEC)***

John P. Rouillard, University of Massachusetts, Boston

All forms of log analysis result in false reports, but the Simple Event Correlator aims to perform automated log analysis while minimizing errors through proper specification of recognition criteria. Information can be gained from logs by looking for certain patterns. These patterns may appear across multiple logs. Absences of events and relationships between events should also be noted.

Relationships between events may take the form of one event taking place before or after another, or in a sequence. Events may also be coincident within a certain period of time. If there is some event that occurs periodically, its absence would be significant.

Relationships between events are important for combining events. For example, some program may emit an error without a username attached to it, but only after another related event that contains a username. Correlation may take place by watching for one event, then expecting another within a certain amount of time that can be matched with the first.

SEC allows for analysis of logs in real time using simple rules for single events, as well as many complex rules which support

matching multiple events within given windows or thresholds and as pairs. With proper use of these rules, relationships between events such as those mentioned above may easily be detected.

INVITED TALK

- *A New Approach to Scripting*

Trey Harris, Amazon.com

Summarized by Josh Whitlock

People think of scripts being different from programs. People think of Perl, Python, and bash shell code as scripts, while they think of C, C++, and Java code as programs. Scripts are interpreted, while programs are compiled. In fact, scripts are a subset of programs. They are programs that make heavy use of their environment (files and directories), define few complex data structures, have no outer event loop, are run by either the author or administrator, and have a primary purpose of ensuring a given state is achieved in a system. Scripts should be distinguished from programs because good programming methodology is well researched while good scripting methodology is not.

As an example, consider a script that wants to mount a remote file system via NFS. If the script does nine things, there are nine places where the script can fail. Restarting the script when it fails could result in unwanted and incorrect duplication of tasks (e.g., multiple mountings). Adding error checking code to the script simply convolutes the code. The problem with error checking code is that it is syntactic and not based on implementation. The error checking needs to be semantic. The Commands::Guarded command set for Perl is available from CPAN. Each guard consists of two parts: a condition, or guard statement, and an executable statement. For example:

```
ensure { $var == 0 }
using { $var = 0};
```

means "if $var is not zero, then set $var equal to zero." To effectively use guarded commands, decompose the code into atomic steps. For each step, write the necessary and sufficient condition for the step to complete. Use the guards as the conditions for each step.

There are many benefits to using guarded commands. They make scripts more resilient because they make testing semantic. If the environment changes, the script is more likely to continue to function.

Guarded commands reduce the amount of error-checking code, thus making the code easier to read and maintain. If a script terminates prematurely, the script will pick back up exactly where it left off, thanks to the guards.

GURU SESSION

- *AFS*

Esther Filderman, The OpenAFS Project

Summarized by Peter H. Salus

Esther (Moose) Filderman knows more about AFS than anyone else I've ever encountered. She illustrated this knowledge and her quick wit for about an hour, at which time the session dissolved into a sequence of queries from the floor, comments, and idle remarks.

She began by noting that "AFS was rather clunky and difficult to administer, but then the universe changed." What happened, of course, was that it moved from Carnegie Mellon to a startup called Transarc, which open-sourced "Andrew."

Originating in 1986, the core has been maintained and guided since 1988 (when Moose joined the crew). It is now "tightly secure" with a Kerberos-like mechanism. "AFS is a good complement to Kerberos," Moose said. It has the structure of a distributed file system, and users can't tell what's what: everything looks like /afs.

There are lots of protections available, but AFS is still "slow," though "speed is coming along nicely." The biggest problem (of course) is with the I/O. "Read times are better; write times are getting better."

Apparently, when there are lots of path or sharing changes, one should opt for AFS over NFS.

Definitely interesting and worthwhile.

SYSTEM INTEGRITY

Summarized by John Sechrest

- ***LifeBoat: An Autonomic Backup and Restore Solution***

Ted Bonkenburg, Dejan Diklic, Benjamin Reed, Mark Smith, Steve Welch, and Roger Williams, IBM Almaden Research Center; Michael Vanover, IBM PCD

LifeBoat is a backup solution designed for ease of use. The goal is to reduce the total cost of ownership for a system by reducing client support costs.

Over 50% of current support costs involve PC clients. While these PC clients often have mission critical software on them, the underlying server automation generally does not reach down to the PC clients. LifeBoat is aimed at resolving this by creating a complete rescue and recovery environment.

Targets for the backup system can be network peers, dedicated servers, or local devices. This means that the backup must include file data as well as file metadata. In order for it to be user friendly, it must enable users to restore individual files. More critically, it must support special processing for open files locked by Windows OS. LifeBoat provides a kernel driver to obtain file handlers for reading locked files.

In order to manage the backed-up data, LifeBoat uses an object storage device called SCARED that organizes local storage into a flat namespace of objects. The data is

not interpreted; it can be encrypted at the client and stored encrypted on the storage devices. These storage devices authenticate clients directly, supporting a method of keeping the data pathways secure.

LifeBoat can use centralized servers, peers, or local devices for backup. This offers some flexibility in the overall problem of finding a good place to put data when you are on the road with your laptop.

LifeBoat is packaged as a Linux boot CD providing software used for maintenance. This allows for a rescue and restoration of a system when there is a problem.

While perhaps the term "autonomic" is being overused in this case, the system looks as though it enables users to easily back up systems, including mobile systems. This is one of the great problems in an organization supporting PC systems. And the solution looks as though it will integrate well in a corporate environment supporting autonomous server configuration strategies.

- ***PatchMaker: A Physical Network Patch Manager Tool***

  Joseph Crouthamel, James Roberts, Christopher M. Sanchez, and Christopher Tengi, Princeton University

  What do you do when you have lots of networks and wires, with 1500 hosts, 675 switch ports, and 1100 patch ports, and you want to keep track of what is plugged into what? Often a huge amount of the time spent debugging a problem involves locating the port causing the problem. Hand-tracing old wires is very time-consuming.

  There used to be a Perl CGI for patch information, but it did not scale very well. As the port count increased, they felt they needed a better tracking system. By putting a Web interface on the front end, it became more useful for a broad group of people.

  This program has a patch database, which is searchable. This is support port monitoring and management and enables direct changes in the VLAN information. It allows for the cables to be documented and for the switch to be managed directly and to track information about each host. It makes it more effective by presenting a visual view of a patch panel.

  This is an open source package written in mySQL, PHP, DHTML, and CSS. It supports SNMP/Sflow and uses Mrtg/nMon to monitor network activities.

  They keep information on patches, racks, hosts, panels, port count, SNMP, and more.

  In the future, they hope to add GUI creation, user authentication and privilege levels, change logging, QoS/rate limiting, security ACL management, switch configuration, file management, and end-user PortMaker.

  This project looks like a local solution that is trying to evolve into a set of broader network management tools. That the whole site can be managed through managing the configuration is an important step forward toward bringing automation to network management. It is too bad that they did not then integrate this data and structure with a configuration management system. When hosts are deployed by a management system, you need to deploy the network as well. This package will work for people to patch by hand, but it has not progressed to the point where a content management system could do it.

  http://www.cs.princeton.edu/patchmaker

- ***Who Moved My Data? A Backup Tracking System for Dynamic Workstation Environments***

  Gregory Pluta, Larry Brumbaugh, William Yurcik, and Joseph Tucek, NCSA/University of Illinois

  NCSA has a lot of highly mobile computers, employees, and students. With increased laptop use, in particular, they found that they had to rethink their approach to data management.

  Far too many people look at laptops as a way to provide persistent data storage, and this puts the data at risk. As the systems move in and out of the network, the backup process can easily fail.

  NCSA started to look at the percentage of laptops vs. percentage of backup success rate. They wanted to be able to find the important data that was vital to the organization.

  They set up a backup tracking system, with authentication servers. This integrates with a Web-based application to go through the data and search for material. It makes a list of machines and users who have not been backed up and allows them to work to increase the backup success rate.

  This talk clearly shows that for organizations that have information as their main product, it is vital to work out a meaningful backup process. And for the most part, people do not back up PCs and they don't back up laptops. NCSA's system provides an organizational framework to enable these backups. This was a good measure for NCSA, but it leaves me wishing for a good distributed file system that knows what to do with mobile users.

SPAM MINI-SYMPOSIUM

- ***What Spammers Are Doing to Get Around Bayesian Filtering & What We Can Expect for the Future***

  John Graham-Cumming, Electric Cloud

  Summarized by Jimmy Kaplowitz

  Graham-Cumming started by describing several spam-related trends that have occurred in the past year or so. Most major email clients support adaptive filtering, with the notable exception of Outlook. Spam is getting slightly sim-

pler and less tricky, but not very fast. Spam is also using Cascading Style Sheets.

The speaker then described specific tricks spammers are using, ranging from hard-to-read color and Internet Explorer's odd handling of hexadecimal color codes to Web bugs and nonexistent zero-width images. From July 2003 to April 2004, email software firm Sophos has noted the relative frequencies of certain tricks. Spammers try to hide red-flag words and include innocuous words. About 10% of spam uses obscure Web site addresses, 20% uses simple trickery such as the insertion of spaces or punctuation characters into red-flag words, and another 20% inserts HTML comments to break up red-flag words. About 80% of spam uses trickery of some sort, but the percentage that doesn't is increasing. Even some bulk email software manuals point out that anti-filtering tricks often make spam easier to filter. There are even spam-sending programs that include SpamAssassin technology to allow the spammer to test and tweak their mail for maximum penetration. A very common occurrence in spam is for the text/plain MIME part to contain very different content from the text/HTML MIME part.

The speaker presented seven tough questions to ask vendors of anti-spam software. The first question is, "How do you measure your false positive rate?" Claims of 99.999% accuracy are meaningless, since a mail filter that deletes all mail can claim to have removed 100% of all spam. Ask what the false positive and false negative rates are, and how the vendor knows. Another question is, "How often do you react to changes in spammer techniques?" The third question, designed to cut through the hype, is, "What are your top two ways of catching spam?" In order to prevent spammers from receiving feedback on your reading preferences, another important behavior to confirm is that the software prevents Web bugs from firing. It also must properly handle legitimate bulk mailings as non-spam, deal gracefully with user reports of legitimate mail as spam, and have a good method to separate non-English spam from other non-English mail.

At the end of the talk, Graham-Cumming predicted that spammers will continue to reduce their use of obfuscations and other trickery and that the set of popular spam tricks will continue to change.

PLENARY SESSION

- ***A System Administrator's Introduction to Bioinformatics***

Bill Van Etten, The BioTeam

Summarized by John Sechrest

BioTeam is an organization of scientists, developers, and IT professionals who have an objective—vendor-agnostic professional services aimed at solving large-scale bioinformatics problems for large companies.

Bill Van Etten provided a quick but detailed introduction to genetics and genomics. He then went on to talk about how computing is impacting current biological activities.

HISTORY

- 1866 Genetic theory published—Mendel
- 1869 DNA discovered—Miescher
- 1952 DNA is genetic material—Hershey
- 1953 DNA structure—W&C
- 1959 Protein structure determined—Perutz, Kendrew
- 1966 Genetic code
- 1977 DNA sequenced—Sanger
- 1988 Human Genome Project started
- 2001 Human genome sequenced

GENETICS TRIVIA

- Everything you are is either protein or the result of protein action.
- Proteins are folded strings of amino acids (20).
- Proteins' structure is important.
- Genes are a hunk of DNA that defines a protein.
- There are 3 billion DNA letters (made up of only 4 characteristics) in the human genome.
- Five percent of human DNA contains genes.
- 999/1000 DNA is identical between any two people.
- Human genes are 98% the same as those of a chimpanzee.
- Human genes are 50% similar to those of a banana.

In 1953, it was discovered that DNA structure is a 3D structure in the form of a double helix. In 1968, the DNA code was broken using Sanger DNA sequencing. DNA of all possible lengths from a known starting point is treated. Each strand ends with a radioactive dideoxy nucleotide which terminates the chain. The strands are weighted.

You get something like:

ACTGAGTGAGCTAACTCA
CATTAATTGCGTT

It all happens in a single capillary tube, and the results are read via laser spectrometer. This leads to high throughput sequencing (see http://www.sanger.ac.uk/Info./IT).

And this leads to an explosion of public sequence databases, which, in turn, leads to a large number of computing related activities: genetic mapping sequence analysis, genome annotation, functional genomics, comparative genomics, expression analysis, coding regions, and genetic coding.

The growth of the genetic scientific inquiry has followed the growth in computing power. All of this information is able to be processed through computer-aided data analysis.

There are some interpersonal differences between wetlab people and computer scientists: Wetlab people know that biology is unreliable but think computers work

well all the time. Computer people think that biology works great but that computers have problems all the time.

The half-life of scientific information is five years.

SEQUENCING SEARCHING ALGORITHMS

There are several sequencing searching algorithms, which build and map sequences. These include Blat, Blast, and HMMR (Hidden Markoff Matching => HMM). They involve many pattern-matching approaches. Each tool is particular to each pattern and structure.

BioInformatics is full of problems that are embarrassingly parallel. There is a great deal of data, but it is generally easy to decompose the problem into a number of little problems. This type of problem leads to a great many cluster and grid-style solutions.

But because many of the people who are trying to use these systems are scientists who have a hard time with computing details, they found that a portal architecture was a fabulous step forward.

All the gory details of LSF, Grid Engine, and Condor are really distractions for scientists who are just trying to get data back out.

While clusters are cheap ways to increase computing power, they are often hard to build, manage, and use. It is also difficult for scientists to map computing to computers in the cluster, making it hard to achieve high throughput and high performance .

BioTeam has developed a tool called iNquiry, which is a rapid cluster installer with a persistent graphical user interface. It is built around the Apple Xserver. You can see an example of it at http://workgroupcluster.apple.com. (Ask for a login.)

- It supports the automatic deployment of a cluster for biocomputing, including: network services; DRM—LSF, etc.; admin tools; monitoring tools; biological applications (200).

  The most impressive idea that I got from this talk was that they wrote a DTD to create a command line interface and write an XML description of the command-line which was used to generate Web interfaces for command lines. This hides system from the scientists in an elegant way that has leverage.

  As a serious win on the cool marketing idea, they used an Apple iPod to build the cluster. All the data is on the iPod, and it drives the boot and install process (and the sysadmin gets to keep the iPod).

  While this works well on the Mac OS X-based Xserver, it works less well on some Linux system hardware platforms. It uses a system imager.

  The core work for the DTD/XML solution was done by Catherine Letondal. It is called pise:

  - http://www.pasteur.fr/recherche/unites/sis/Pise/CCP11/s6.html
  - http://www.pasteur.fr/recherche/unites/sis/Pise/
  - http://bioinformatics.oupjournals.org/cgi/reprint/17/1/73.pdf

  For more details that might interest you, read Bill Bryson's book *A Short History of Nearly Everything*.

  Web site: http://bioteam.net

SECURITY

Summarized by Tristan Brown

- ***Making a Game of Network Security***

  Marc Dougherty, Northeastern University

  The idea is to make a closed, firewalled network with a collection of hosts running various unpatched and insecure services. Give each team a host, and give the teams two days to defend their machines while exploiting everyone else. The result is Capture the Flag tools, a framework for online war games. Run services on your host and gain defensive points. Compromise other people's servers and gain offensive points. The toolkit facilitates the competition in two ways:

  1. Service verification. A flag service is run on each host, allowing the central server to perform confirmation of the actual flag file on the server. The process allows the central server to determine what team is in control of each server and to defeat simple cron jobs designed to ensure that one team's flag file stays in place. This is a complex procedure and is currently being overhauled to become even more robust.
  2. Scoring. A scoreboard showing each team's progress is kept available. This simple touch is an effective motivator to keep people working on the project.

  The tool framework has room for future expansion. Different games can be implemented (verification is based on Perl scripts), and a larger game based on a large network of networks or autonomous systems is being planned. More information can be found at http://crew.ccs.neu.edu/ctf/.

- ***Securing the PlanetLab Distributed Testbed: How to Manage Security in an Environment with No Firewalls, with All Users Having Root, and No Direct Physical Control of Any System***

  Paul Brett, Mic Bowman, Jeff Sedayao, Robert Adams, Rob Knauerhause, and Aaron Klingaman, Intel Corporation

  PlanetLab is an environment with unique security requirements—the system must remain secure despite the fact that owners and users of each system have root access on the machines. To ensure availability, several tools are used. Nodes run virtual servers to allow users to operate as root in their virtual server but not in any oth-

ers. Packets are tagged to identify the originator of all data, thus ensuring accountability among the research-ers. Traffic control is utilized to ensure that no one node uses excessive system bandwidth. In the event of a compromise, systems can be remotely rebooted with a magic packet, causing them to start off of a CD. The machine then enters debug mode, restricting access and allowing remote forensics to be performed. Patches may be applied, bringing the node back to a known, safe state.

Because of all the measures taken to protect the network, a large compromise resulting in rooted nodes was detected and contained within minutes. Further security reviews since have resulted in an effort to decentralize administration and eliminate reusable passwords. To support future expansion, a federated decentralization scheme is being implemented.

- ***Secure Automation: Achieving Least Privilege with SSH, Sudo, and Suid***

  Robert A. Napier, Cisco Systems

  Automation and scripting across the boundary between hosts is a difficult process often subject to security vulnerabilities. Although insecure tools such as Telnet and RSH may no longer be used, SSH and sudo can still be used to exploit a system. A design paradigm of small, simple scripts that do one task very well and with the least privilege possible can help to prevent unauthorized access due to an improperly written script. Several techniques can be used to harden scripts:

  - SSH command keys can be used to perform a command on another host without the possibility of using the session to execute a shell. This ensures that a script can only perform its single task on the remote machine, and it keeps attackers from performing unwanted actions on the remote machine.
  - Properly configured sudo can give scripts the ability to do just what they need and no more. It should not be used to give scripts complete root access.
  - Scripts should have a unique user ID to isolate their domain on the system even further.
  - Setuid and setgid can both be used to allow a script to work as another user. Setgid has the advantage that fewer exploits have been performed against it. Setuid to root is generally a bad idea for a script.

  By following the principle of least privilege and asking, "How much access do I need?" rather than "How much security do I need?" overall vulnerability due to scripts can be reduced.

INVITED TALK

- ***System Administration and Sex Therapy: The Gentle Art of Debugging***

  David Blank-Edelman, Northeastern University

  Summarized by Peter H. Salus

  Blank-Edelman's underlying thesis is that we can improve our system administration through learning from other fields. He likes sex therapy. This is because:

  - Debugging is getting harder.
  - Debugging is not merely binary.
  - Both fields (sysadmin and sex therapy) deal with complex systems.

  The phenomena are harder because of interdependency; because we don't control the "parts" and haven't written the software; as availability increases, the level aided decreases.

  We all feel that "sex should just work." Blank-Edelman listed a number of male and female myths concerning sex that he found in several therapy books. He then moved to "system administration should just work," where the principal assumptions involve:

  - Plug and play
  - It's just (merely)…
  - No administration toolkit
  - Printing
  - The Internet
  - "You have the source, right?"

  Blank-Edelman analogized between sex therapy and Agans' (2002) "Nine Indispensable Rules":

  1. Understand.
  2. Make it fail.
  3. Look at it.
  4. Divide and conquer.
  5. Change one thing at a time.
  6. Keep an audit trail.
  7. Check the plug.
  8. Get a fresh view.
  9. If you didn't fix it, it isn't fixed.

  Good advice.

  The comparative metaphor was attractive and Blank-Edelman is a good presenter, but it wore thin long before the 90 minutes were up, and there were too many old and trite jokes and anecdotes.

GURU SESSION

- ***RAID/HA/SAN (with a Heavy Dose of Veritas)***

  Doug Hughes, Global Crossing; Darren Dunham, TAOS

  Summarized by Andrew Echols

  The RAID/HA/SAN guru session consisted of a Q&A, primarily about using Veritas products for RAID, high availability, and SAN setups.

  The gurus recommended a few mailing lists for questions related to the session topics. Each list is at <list-name> mailman.eng.auburn.edu, and uses <list-name>-request mailman.eng.auburn.edu to subscribe:

  - Veritas Applications: veritas-app
  - Veritas Backup Products: veritas-bu

- Veritas High Availability Products: veritas-ha
- Veritas VxVM, VxFS, and HSM: veritas-vx
- Sun storage appliance: ssa-managers

A small sample of the topics discussed follows.

Regarding experience with VxVM and Sun Cluster, it was noted that Solaris 10 has a new file system, ZFS, which has its own volume management. There are known issues, but Sun is likely motivated to drop the requirement for VxVM to use Sun Cluster. Sun is also pushing ZFS as something that just does the right thing and hides the details.

In another case, a user recently started using Veritas Foundation Suite 4 and wanted to reduce the number of inodes. There appeared to be a very high amount of overhead, but it was suggested that there might be differences between definitions of gigabytes (2^30 vs. 10^9) between programs. It was also noted that there may be problems with the size of the private region.

Commenting on the maturity of the Linux Volume Manager, the gurus felt that it had some nice features, primarily a combination of those provided by Veritas and AIX. The naming scheme is good and fairly robust. It still needs mirroring, but that can be worked around at the RAID level.

THEORY

Summarized by John Sechrest

- ***Experience in Implementing an HTTP Service Closure***

Steven Schwartzberg, BBN Technologies; Alva Couch, Tufts University

This talk is a part of an ongoing discussion in configuration management. How can you create a system which can programmatically be configured in a coherent and consistent way?

Two concepts put forward are "closures" and "conduits." Closures act as an element of the network and conduits are the mechanism by which closures talk to each other.

A closure is a self-contained application that is self-managing and self-healing and where all operations affecting the closure must be within this or another closure.

If this is so, then the closure will:

- Provide consistency
- Create a common command language
- Operate the same way regardless of OS or other applications
- Provide security
- Provide a single interface into the closure
- Detect intrusion and potentially repair modifications
- Hide complexity
- Understand the dependencies and be a mediator
- Provide configuration language in plain English
- Eliminate typographical errors

All of these things are good ideas.

This was proposed in an earlier paper, but how does it work in reality? This talk covered the experience of building an HTTP closure:

1. Start with RH 9 w Apache.
2. Create a protected directory.
3. Wrap Apache binaries with scripts.
4. Move all HTML, logs, configuration files, and modules into the repository.
5. Create a Perl script to allow management of the closure.

At the first iteration of the closure language, they supported a limited command set:

- assert—Create a virtual domain.
- post—Upload a file into the domain.
- retract—Remove a file from the domain.
- allow—Grant permission to users and modify configuration options.
- Deny—Remove permissions as above.

For example:

```
Assert foo.edu
post info.html
www.foo.edu/info.html
allow cg www.foo.edu/cgi-bin
retract www.foo.edu/
information
```

But this leads to problems and ambiguity. They had problems with how they specified files vs. directories, the renaming of files, and dealing with indexes.

In order to address these problems, they tried to gain idempotence (making order not matter) by hiding the complexity of the commands and trying to create statelessness. Post must only deal with directories, so you load the whole site each time.

This was an interesting experiment. It showed that there were many places where procedural complexity and all the details of a configuration make it hard to present a simplified solution. This was a good first step in understanding how a closure might be built. However, Apache seems like a difficult example to process. Perhaps a simpler service would be a better starting point for illustrating the idea of closure.

- ***Meta Change Queue: Tracking Changes to People, Places, and Things***

Jon Finke, Rensselaer Polytechnic Institute

At RPI all of the data about all of the different systems is stored in databases.

Getting this data where it needs to go, and doing it in a timely manner, has taken some thought. Many of the issues are related to who owns the data and who needs the data. Often these are in different parts of the organization.

Different data sets—CMMS, LDAP, ID card, Space—need to be integrated into the process. By setting up a Meta_Change_Queue Table, the changes to the data sets can be identified. A Listener can be set up to catch and process these data changes in a low-overhead way.

This table uses a vendor-provided trigger to input real-time data. This protects the business logic of the program, which allows third-party programs to work completely and still be integrated into the larger campus data process.

Through this mechanism a single sign-on system for campus has been created and maintained.

In order to do this, Jon needed to:

- Index a key column and set to null after processing
- Import with a trigger, which works well with Oracle target systems
- Include a low-priority batch queue

A key idea is to use a trigger to catch deletions and move things to the history table on deletes.

All source code is available on Jon's Web site (www.rpi.edu/~finkej/), but it is not packaged and is mostly in PLSQL.

- ***Solaris Zones: Operating System Support for Consolidating Commercial Workloads***

Daniel Price and Andrew Tucker, Sun Microsystems, Inc.

Solaris Zones are a way to run multiple workloads on the same machine. They are akin to Jails in BSD. Because they all use the same kernel, they gain efficiency, but they do not gain the independence of multiple kernels like User Mode Linux. These are not virtual machines.

A set of processes look the same at the base machine. It supports a reduced root-permissions structure, in order to prevent escaping from a zone as you can from a chroot.

They support per-zone resource limits.

This was an interesting introduction to Solaris Zones. However, it did end up coming off as a sales talk more than a comparison with the state of the art. Solaris Zones are a good thing, just as Jails are a good thing. I wished there had been a broader perspective in this talk.

INVITED TALK

- ***Used Disk Drives***

Simon Garfinkel, MIT Computer Science and Artificial Intelligence Laboratory

Summarized by Jimmy Kaplowitz

Simon Garfinkel explained that much data is frequently left behind when a computer owner tries to remove it from the hard drive. This is because of the common choice to use quick but vastly incomplete erasure methods such as the DEL and FORMAT commands of Micro-soft operating systems. The data left behind by these tools stays around for a long time, since hard drives are quite long lived. Physical security and OS-level access controls fail to ensure data privacy when drives are repurposed or given to a new owner. Cryptography, which would provide that assurance, is rarely used on-disk.

The author did a study of 235 used hard drives, some of which were obtained for as little as $5. The contents of the drives included runnable programs and private medical information. The data was deleted to different degrees, so the author developed several different levels of deletion to keep these separate. Level 0 files are ordinary files in the file system. Level 1 files are temporary files in standard temporary directories. Level 2 includes recoverable deleted files, such as those in DOS where only the first letter of the filename is lost. Level 3 is partially overwritten files.

An overview of digital forensics tools was given. Among hard-drive forensics tools, the two main categories were consumer and professional tools. All of them were able to undelete files at level 2 and search for text in level 3 files. The professional tools also had knowledge of file formats, were able to perform hash-based searching, and could provide reports and timelines useful for auditing and legal testimony purposes.

The author then described the challenge of performing forensics on many drives at once with very little time for each drive, explaining some of his choices of tools as well as some of the gotchas he had to overcome. In total, he worked with 125GB of data. The data included very common parts of the OS, Web caches, authentication cookies, credit card numbers, email addresses, medical records, personal and HR correspondence, personnel records, and similar items.

Garfinkel told several more horror stories about really sensitive data being revealed, and suggested the creation of a US privacy commissioner or some other federal waste-auditing role. He cautioned that all of his warnings about recoverable data on hard drives applies equally well to USB drives, digital cameras, and other types of media. However, when faced with a police officer ordering deletion of (for example) a picture on a digital camera, the recoverability of dirty deletion is usable to preserve one's data.

The author mentioned the Department of Defense's standard for sanitizing media containing non-classified data and then proceeded to discuss other tools for overwriting media. The tools discussed range from standard UNIX dd to commercial tools such as Norton Disk Doctor. It was pointed out that Windows XP and NT hide a little-

known sanitization program in their installation; it's called cipher.exe.

The author discussed the exotic threat of hard drives with hostile firmware that lie about their state and snoop on other hard drives in the same system. It is even possible for such hard drives to infect other drives with hard-drive viruses. There is a level 4 part of the disk, the vendor area, which is where the firmware lives. This can be overwritten by really expensive sanitization tools.

The talk concluded with a sobering question: If the author was able to get so much private info for under $1000, who else is doing this?

- ***Work-in-Progress Reports***

  Summarized by John Sechrest

  WiPs are brief introductions to topics that people are currently working on. Seven people stood up and gave rapid descriptions of ongoing projects.

  - *OGSAConfig*—A proof of concept implementation of an architecture to make grid fabrics more manageable

    Edmund Smith, University of Edinburgh

    Split the problem down for a grid by allowing each node in the grid to get a different profile. You can do this with a constraint-resolution problem.

    OGSA config project: http://groups.inf.ed.ac.uk /ogsaconfig/

  - *CfengineDoc*—A tool which uses GraphViz and NaturalDocs to document the import execution order of cfengine scripts

    Christian Pearce, Perfect Order

    How to describe the meaning of a cfengine file. Used Inkscape to draw a cfengine config file and then discovered GraphViz. Then got AutoDoc and found NaturalDocs. Use it to describe SysNav Interface.

    http://cfenginedoc.com
    http://naturaldocs.com
    http://autodoc.com
    http://cfwiki.org
    http://sysnav.com
    http://inkscape.com

  - *Verifiable Secure Electronic Voting*—An electronic voting system that allows voter verification of individual votes, and makes a passing nod towards being a secure and private voting environment

    Zed Pobre, Louisiana State University

    Use a UID + voting ID + MD5. Supports many different voting styles.

  - *Grand Unified Documentation Theory*, or What's wrong with current documentation standards. Can we fix it?

    David Nolan, CMU

    Trying to create an interface to a single place for documentation using a wiki front end, with some customizations for raw access and a choice of editor.

    vitroth+lisa04@cmu.edu
    sadm-l@lists.andrew.cmu.edu

  - *CADUBI: Creative ASCII Drawing Utility by Ian*

    Ian Langworth, Northeastern University

    Received Best WiP Award!

    CADUBI is in all kinds of ports. It is Old Crap, very old.

    CADUBI 1.2: two maintainers to the last version.

    Go AWAY . . . Use TetraDraw.

    (This was incredibly humorous.)

  - *Geographic Firewalling*

    Peter Markowski, Northeastern University

    Wireless firewalls are natively geographic entities. This way you can choose where people get services. Limit your perimeter.

  - *Portable Cluster Computers and InfiniBand Clusters*

    Mitch Williams, Sandia National Laboratories

    Received 2nd Best WiP Award!

    A cluster in a box. It is one foot tall, has four CPUs, and by using LinuxBios it can boot in 12 seconds. Working on adding infiniband and creating embedded Linux clusters.

INVITED TALKS

- ***Lessons Learned from Howard Dean's Digital Campaign***

  Keri Carpenter, UC Irvine;
  Tom Limoncelli, Consultant

  Summarized by Andrew Echols

  Howard Dean's presidential campaign was unconventional in its approach to putting the campaign online. Traditionally, candidate activities take the form of getting out to the people, giving speeches, and kissing babies. Online campaigning takes this approach further, putting more content online and organizing supporters over the Web.

  Joe Trippi, the Dean campaign manager, a veteran of the dot-com boom, wanted to run an "open source" campaign. Traditional campaigns have been online before, but they have limited themselves to creating, essentially, a brochure Web site. The Dean campaign, however, created an online social movement with tools like blogs, meet-ups, and mailing lists. Furthermore, control of the message was opened up to Web site visitors. Supporters were trusted and ex-pected to aid in crafting the movement.

  The campaign blog provided up-to-the-minute articles and discussion of activities. It provided a central online community for the campaign. It also allowed visitors to post comments, opening up the dialog.

  The potential of the online campaign is apparent when one considers where it took the campaign. In January 2003, the campaign

had only 432 known supporters and $157,000 in the bank. By the end of the campaign, it had raised over $50 million from 300,000 individuals, with 640,000 supporters on a mailing list and 189,000 meet-ups. The online campaign's success story is that it took Howard Dean from obscurity to being, for a time, the front-running contender for the Democratic nomination.

- ***Storage Security: You're Fooling Yourself***

  W. Curtis Preston, Glasshouse Technologies

  Summarized by Josh Whitlock

  Until recently, storage people did not have to worry that much about security. Because storage was not meant to be accessible on a network, security was not designed into storage technology. Now that network security is becoming more of an issue, storage security must be reconsidered. One problem is that storage and security people speak different languages. Storage people talk in terms of zones, LUNs, and mirroring, while security people talk in terms of ACLs, VLANs, and DMZs. By learning how each other operate and working together, storage and security people will be better able to locate and address vulnerabilities.

  There are a number of security issues for storage. People can get inside the SAN and there have full access to the data by exploiting configuration mistakes such as open management interfaces and default passwords. Management interfaces are often plugged into the corporate management backbone, are visible from the LAN, use plaintext passwords, and often have default passwords. This problem can be solved by putting management interfaces on a separate LAN and upgrading to interfaces like SSL and SSH that don't use plaintext passwords. The backup person can be the greatest security risk of the SAN, because that person has root access. Depending on the backup person's security astuteness, a hacker could gain access to the backup server via social engineering and insecure server configuration.

  A hacker with access to the backup server has complete control over all the data. The greatest threat to tape backups is theft. Social engineering could be used to sway a low-income employee transporting the backups into stealing the tapes. Discarded media that is not sanitized properly before being sold could be scavenged for data. SAN issues with security deal with zones. For enforcement methods, hardware-enhanced zones are the only type of zoning that offers a meaningful level of access control. LUN masking hides LUNs from specific servers and should not be considered a security mechanism, but commonly is.

  Suggestions for improving storage security include shutting down whatever is non-essential, testing NAS servers for weaknesses, and putting the NAS on a separate LAN. Learning about weaknesses, planning for security, and working with vendors to make storage technologies more secure are other means of improving security.

IMPRESSIONS FROM A LISA NEWBIE

Chris Kacoroski, Comcast
kacoroski@comcast.net

Hi,

As a LISA '04 newbie with a focus on configuration management training, I attended tutorials every day and missed many of the technical sessions. What follows are my opinions, which you are obviously free to agree or disagree with.

Overview:

Having all areas covered by wireless really changes how the audience listens to the speakers. I would go to a session and while the speaker was setting up, I would go online and check out the servers at work, along with my favorite Web sites. I noticed several other people also doing this. If the speaker was covering something I was not interested in, I responded to emails. With the IRC channel, I saw some folks holding side chats while speakers were talking. This is a good thing for folks like me who do not have a backup in their job (one day I was able to fix a problem with our name servers), but from a speaker's viewpoint, I would think it could be very frustrating that some people are not paying atten-tion.

I was impressed by LISA's leadership and staff's emphasis on learning from each other. They encouraged this via the LISA bingo game, where you had to get signatures of people, the lunches provided with the tutorials, and the acknowledgment of the importance of "hallway track," where I definitely learned the most. I was also impressed by the folks who taught the tutorials; these are people who write books (e.g., O'Reilly's *LDAP* by Gerald Carter).

The days were very intense, with sessions from 9 to 5:30. Then there were BoFs from 7 to 11 p.m. After that the hallway track typically continued from 1 to 3 a.m. I averaged about five hours of sleep a night. I definitely had information overload.

The high points for me were:

- *System Logging and Analysis Tutorial*

  Marcus Ranum

  I liked the idea of "artificial ignorance," where if you see a log message that is okay, you filter it out but keep a count of how often it happens. In a short period of time, the only messages that get past the filters are ones that you have not seen before and that you need to look at. By keeping a count of messages, you can be alerted if a

message that normally happens 100 times a day jumps to 1000 (something probably happened). I also learned just how unreliable UDP really is and ways around this problem (using TCP or local server filtering).

Marcus briefly mentioned logwatch, plog, and logbayes, and went into detail on his NBS tool, which is used to implement the idea of Artificial Ignorance. Marcus is a good presenter who makes the material real through several real-life examples.

- *Cfengine Tutorials*

  Mark Burgess

  The introductory tutorial was not that interesting, but the advanced tutorial was excellent. I picked up enough new ideas and hints that I really need to go back and re-read the manual (some of this is due to a new version that has some significant changes, such as subroutines). Mark even created a patch to re-solve a specific problem I have been having with my use of cfengine.

- *Change Management Guru Session*

  Gene Kim

  The book provided with this session is very good, as it provides a recipe that my management can use to start to implement good change-management practices. Because this book is based on real data about the impact of unmanaged changes, I think I will have a good chance of getting it implemented.

- *Configuration Management Session*

  Alva Couch/Mark Burgess

  Mark's discussion on computer immunology did not grab me—too far out in the future, but Alva's discussion about the phases of configuration management and the difficulty in moving from one level to another was excellent, as he also listed the benefits of moving to the next level. I will be able to use this to show my management where I am trying to go. The phases are:

  Level 0—Ad hoc: Uses ad hoc tools to manage the systems. Pretty much anything goes. There are no barriers, since everybody does their own thing. Also, there is no documentation, reproducibility, or interchangeability of systems. This starts to fail when the number of systems goes above 50–100.

  Level 1—Incremental: Uses incremental management—implements a tool such as cfengine and just manages a few things on the machines. One barrier is that each sysadmin has to be trained on the tool you use. This level provides documentation of the systems and ease of updating 1000s of systems. This starts to fail when you need to manage entire systems, not just parts of them.

  Level 2—Proscriptive: Controls the entire system by building from bare metal for each system. A barrier is loss of memory of how a machine was set up (e.g., why a machine was configured in a certain manner). Lots of work to document exactly how the system was set up before it was rebuilt. Need to do a lot of testing to ensure that the bare-metal rebuild will work. This provides guaranteed reproducibility, where you can create any system from bare metal.

  Level 3—Enterprise: Divorces the hardware from the machine use so machines are interchangeable—we can move services between machines as needed. All machines are created alike. A barrier is the loss of freedom to take care of your own machine (e.g., you do not know which machine will be the Web server). The benefit is that any machine is interchangeable with another. Kind of like thin clients, where a person can log on to any terminal and get their own environment. Companies like CNN use this for their Web server farms.

  Alva then went on to show how each organization will need to determine which level has the best payoff. For example, his group has decided to stay at level 2, because they rebuild machines every six months and it makes no sense to go to level 3. Paul Anderson's group has moved to level 3, because they do not rebuild machines every six months, so the cost-benefit equation makes sense.

- *BoFs & Hallway Track*

  The asset management BoF had the idea of tracking MAC address to IP address to switch port in order to alert you to any changes in the data center. I really liked what one participant had to say: he used SNMP to get all kinds of data from his servers—the only thing he was lacking was an automated process to determine what power plug a computer was using.

  As I said at the beginning, I focused on configuration management. My environment consists of about 100 servers (Mac, Linux, Solaris, Windows), 1500 need-to-be-managed Mac workstations, and 700 Windows workstations. What I learned from the BoFs, hallway track, and sessions was:

  - Cfengine is a good choice for me to manage my systems.
  - I need to move to automated builds of systems.
  - Cfengine has a lot of functionality that I was not aware of. I had been trying to figure out hacks for certain items that cfengine does out of the box.
  - Many folks who use cfengine do not really use it in depth but only use a small portion of it. Mark Burgess was the only person I found who really understood all of it.
  - I have about a two- to three-year project ahead of me to get everything managed.
  - A monitoring system that uses SNMP to tie MAC address to IP address to switch port has lots of benefits for asset management and

sending alerts when things change.

- Change management is critical. Tracking the success rate of approved changes really makes it much easier to diagnose and resolve problems, since most problems result from recent changes.

cheers,

ski

EuroBSDCon 2004

Karlsruhe, Germany

October 29–31, 2004

Summarized by Jan Schaumann
jschauma@netmeister.org

EuroBSDCon 2004, the third installment of BSDCon in the Old World, opened with a full day of tutorials, which was followed by two days of presentations concluding with the Best Paper award.

Speakers were unfortunately not able to sit in on the tutorials, so instead of learning about "Debugging Kernel Problems" from Greg Lehey, "Using IPv6 on BSD: Concepts, Configuration and Operation" from Benedikt Stockebrand, "Hands-on CVS" from Albert Mietus, or "Content Management with TYPO3" from Martin Alker and Ursula Klinger, I took advantage of the free wireless network and spent most of the day preparing for my own talk.

Jordan Hubbard's keynote speech, "*BSD Is Dying—Anonymous Coward, Slashdot, ©1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004," on Saturday morning marked the official start of the conference. As manager of BSD Technologies for Apple Computer Inc., Jordan doesn't require much introduction in the BSD community. He stressed the fact that with Mac OS X being based on BSD, this family of operating systems has quickly become the biggest desktop UNIX variant. (Judging by the count of Apple laptops among the audience, it seemed that Apple's hardware also enjoyed some popularity among BSD hackers, though, of course, several people were running their favorite BSD in favor of Mac OS X.)

The 23 papers presented were divided into two tracks each day, forcing attendees to make often difficult decisions when choosing among two concurrently scheduled topics. It might be worth mentioning that NetBSD—often playing a smaller role than its cousins when it comes to media exposure—dominated the conference, with a total of 11 papers presented either by NetBSD developers or on NetBSD-related topics.

The first talk I attended was Antti Kantee's presentation on "Using Application-Driven Checkpointing for Hot Spare High Availability," in which he discussed a kernel interface for checkpointing processes, separating them into data and metadata, and making it an application's responsibility to make sure that checkpoints are taken rather than relying on an outside process. This approach allows time-critical applications to provide high reliability on a software level much in the way that RAID provides redundancy and data security on the hardware level.

Having discussed "The Flaf Filesystem" with the paper's author, Søren Jørvang, the night before, I decided to remain in track "A." The "first load all files" file system presented in Søren's paper uses an interesting approach: At mount time, all metadata for the file system is loaded into memory. This reduces the code size enormously, but, at the same time, provides modern features like crash robustness and high metadata performance by taking advantage of ever increasing memory size and disk bandwidth on today's systems compared to average file system usage.

In the next session, Alistair Crooks (in the first of three talks) and Valeriy Ushakov discussed issues related to porting NetBSD to handheld devices. The geek appeal of seeing live demonstrations of an HP Journada executing a bootloader from within its native operating system, Windows CE, to boot into NetBSD is certainly nonnegligible. Furthermore, this talk, consciously or not, prepared the audience for the following two talks in track "A": As handheld computers have limited resources, getting third-party applications installed is not always easy, but the NetBSD Packages Collection may promise some remedy, as was shown in the first afternoon sessions.

Alistair Crooks' second talk focused on the NetBSD Packages Collection and its strength as a portable framework, followed by Krister Walfridsson's presentation on "Cross-Building Packages" from a powerful machine for another architecture. "A Portable Packaging System" provides a nice overview of the NetBSD Packages Collection, or "pkgsrc," as it's known by its users. In it, Alistair focused on the portability, and the problems encountered during the porting, of the framework, which allows system administrators of such different UNIX flavors as Darwin, Solaris, IRIX, and even Interix (to name a few) to manage over 5000 third-party applications.

As mentioned above, building large packages from source may take
significant resources or time on certain platforms. The NetBSD operating system is fully cross-compilable, yet taking this approach in the third-party software management framework is challenging, to say the least. Krister Walfridsson has developed an approach to allow for just that, by intercepting the exec system calls and modifying "the commands in such a way that they do the corresponding operation for the target

architecture." His presentation included a live demonstration of this remarkable system.

Track "B," at the same time, saw a talk by Dirk Meyer on "Lightweight FreeBSD Package Cluster in a Jail," a paper that was to win the Best Paper award of the conference. As I said, it was difficult choosing which talks to attend.

Martin Husemann, scheduled to present his paper "Fighting the Lemmings," unfortunately could not attend EuroBSDCon due to illness. On short notice, Dirk Meyer filled in with a demonstration of "Making Life Easy with FreeBSD Filesystem Snapshots." This much more practical session was welcome after the largely theoretical day.

Saturday concluded in Track "A" with Dru Lavigne's presentation "But I Am Not a Developer…How Can I Contribute to Open Source?" Even though Dru struggled with some technical difficulties and could only project parts of her slides, this talk was very well received and won her the award for second-best paper. In it, Dru shared thoughts and insights from a much less technical point of view, which certainly was appreciated by the mostly developer-focused conference audience.

The next morning, talks started at 9:15 a.m. with Hubert Feyrer's talk about "vulab," a system Hubert developed to enable students to perform exercises as part of a course in system administration he teaches at the University of Regensburg. The system enables the instructor to define certain tasks, set time limits, and even check whether the task has been finished correctly. It seemed unfortunate that due to the early time slot (many people had stayed up late in the hotel bar), only a few people attended.

My own talk, "NetBSD/Desktop: Scalable Workstation Solutions," in the following time slot seemed to be well received, but I should let somebody else be the judge of this. Like the majority of the other talks, the paper and slides can now be downloaded from the conference Web site.

Following the coffee break, I attended Stucchi Massimiliano and Matteo Riondato's "code walkthrough and a case study" of the FreeSBIE system, a LiveCD based on FreeBSD. Stucchi and Matteo won the award for the third-best paper, based on the positive feedback that conference attendants showed for this very practical talk.

"The NetBSD Status Report" is a regular presentation at many conferences (such as USENIX, or more recently, at SUCON '04) and provides a nice summary of where the NetBSD Project stands, where it's headed, and what kinds of new features can be expected in future releases. Naturally, this status report, presented by Ignatios Souvatzis, focused on the upcoming 2.0 release and the release's performance improvements. The NetBSD Project had also released their new logo the night before (incidentally also from EuroBSDCon), but due to the short notice this did not make its way into the status report. After all, Ignatios had to present yet another talk after the lunch break, "A Machine-Independent Port of the SR Language Run-Time System to NetBSD."

The conference talk program concluded with Alistair Crooks' third (but who's counting?) presentation, "The A-tree—A Simpler, More Efficient B-tree." It turns out that Alistair used this talk as an excuse to reminisce about how times have changed since the '70's. The relation here is that obviously times have changed since B-trees were discovered in 1970. Since systems nowadays have an abundance of memory and disk space compared to then, memory-to-memory copying has become much faster, allowing a larger set of information to be held in memory at a time. (You will notice that this is basically the same assumption underlying Søren Jørvang's flaf, as mentioned above, so maybe these guys are on to something here.) The conclusion of his paper is that this approach allows for search times equivalent to those in a B-tree, while insertion and deletion are much simpler.

The conference ended with the closing remarks and the presentation of the Best Paper awards. EuroBSDCon ended officially just in time, and it seemed everybody agreed that it was a big success.

Conference Web site: http://2004.eurobsdcon.org.

Announcement and Call for Participation

19th Large Installation System Administration Conference (LISA '05)

Sponsored by USENIX, the Advanced Computing Systems Association, and SAGE, the People Who Make IT Work

http://www.usenix.org/lisa05/cfp

December 4–9, 2005 **San Diego, California, USA**

Important Dates

Draft papers, extended abstracts, and invited talk and workshop proposals due: *May 10, 2005*
Notification to authors: *June 2005*
Final papers due: *September 27, 2005*

Conference Organizers

Program Chair
David N. Blank-Edelman, *Northeastern University CCIS*

Program Committee
Gerald Carter, *Samba Team/Hewlett-Packard*
Strata Rose Chalup, *VirtualNet Consulting*
Lee Damon, *University of Washington*
Rudi van Drunen, *Leiden Cytology and Pathology Labs*
Joe Gross, *Google*
Jason Heiss, *Overture, a Yahoo! company*
Tom Limoncelli, *Cibernet Corp.*
John "Rowan" Littell, *Earlham College*
Tom Perrine, *Sony Computer Entertainment America*
Yi-Min Wang, *Microsoft Research*
David Williamson, *Tellme Networks*
Elizabeth Zwicky, *Acuitus*

Invited Talk Coordinators
William LeFebvre, *Independent Consultant*
Adam S. Moskowitz, *Menlo Computing*

Guru Is In Coordinator
Philip Kizer, *Texas A&M University*

Workshops Coordinator
Luke Kanies, *Bladelogic, Inc.*

Training Program Coordinator
Daniel V. Klein, *USENIX*

Get Involved!

Experts and old-timers don't have all the good ideas. This is your conference, and you can participate in the planning in many ways:

- Submit a draft paper or extended abstract for a refereed paper.
- Suggest an invited talk speaker or topic.
- Share your experience by leading a Guru Is In session.
- Propose a tutorial topic.
- Organize or suggest a Birds-of-a-Feather (BoF) session.
- Email an idea to the Program Chair.

Overview

The annual LISA conference is the meeting place of choice for system, network, database, storage, security, and all other computer-related administrators. Administrators of all specialties and levels of expertise meet at LISA to exchange ideas, sharpen old skills, learn new techniques, debate current issues, and meet colleagues and friends.

People come from over 30 different countries to attend LISA. They include a wide range of administration specialties. They hail from environments of all sorts, including large corporations, small businesses, academic institutions, and government agencies. Attendees are full-time, part-time, student, and volunteer admins, as well as those who find themselves performing "admin duties" in addition to their day jobs. They support combinations of operating systems ranging from open source, such as Linux and the BSD releases, to vendor-specific, including Solaris, Windows, Mac OS, HP-UX, and AIX.

Refereed Papers

Refereed papers explore techniques, tools, theory, and case histories that extend our understanding of system and network administration. They present results in the context of previous related work. The crucial component is that your paper present something new or timely; for instance, something that was not previously avalable, or something that had not previously been discussed in a paper. If you are looking for ideas for topics that fit this description, the Program Committee has compiled a list of some good open questions and research areas, which appears in a separate section below. This list is not meant to be exhaustive; we welcome proposals about all new and interesting work.

It is important to fit your work into the context of past work and practice. LISA papers must provide references to prior relevant work and describe the differences between that work and their own. The online Call for Papers, http://www.usenix.org/lisa05/cfp, has references to several resources and collections of past papers.

Proposal and Submission Details

Our submission requirements this year are a bit different from prior years, so please take a moment to read these new guidelines. Anyone who wants help writing a proposal should contact the Program Chair at lisa05chair@usenix.org. The conference organizers want to make sure good work gets published, so we are happy to help you at whatever stage we can in the process.

Proposals can be submitted as draft papers or extended abstracts. Draft papers are preferred. Like most conferences and journals, LISA requires that papers not be submitted simultaneously to more than one conference or publication and that submitted papers not be

previously or subsequently published elsewhere for a certain period of time.

Draft papers: A draft paper proposal is limited to 16 pages, including diagrams, figures, references, and appendices. It should be a complete or near-complete paper, so that the Program Committee has the best possible understanding of your ideas and presentation.

Extended abstracts: An extended abstract proposal should be about 5 pages long (500–1500 words, not counting figures and references) and should include a brief outline of the final paper. The form of the full paper must be clear from your abstract. The Program Committee will be attempting to judge the quality of the final paper from your abstract. This is harder to do with extended abstracts than with the preferred form of draft papers, so your abstract must be as helpful as possible in this process to be considered for acceptance.

General submission rules:

- All submissions must be electronic, in ASCII or PDF format only. ASCII format is greatly preferred. Proposals must be submitted using a Web form located on the LISA '05 Call for Papers Web site, http://www.usenix.org/lisa05/cfp
- Submissions containing trade secrets or accompanied by nondisclosure agreement forms are not acceptable and will be returned unread. As a matter of policy, all submissions are held in the highest confidence prior to publication in the conference proceedings. They will be read by Program Committee members and a select set of designated outside reviewers.
- Submissions whose main purpose is to promote a commercial product or service will not be accepted.
- Submissions can be submitted only by the author of the paper. No third-party submissions will be accepted.
- All accepted papers must be presented at the LISA conference by at least one author. One author per paper will receive a registration discount of $200. USENIX will offer a complimentary registration for the technical program upon request.
- Authors of an accepted paper must provide a final paper for publication in the conference proceedings. Final papers are limited to 16 pages, including diagrams, figures, references, and appendices. Complete instructions will be sent to the authors of accepted papers. To aid authors in creating a paper suitable for LISA's audience, authors of accepted proposals will be assigned one or more shepherds to help with the process of completing the paper. The shepherds will read one or more intermediate drafts and provide comments before the authors complete the final draft.

For administrative reasons, every submission must list:

1. Paper title, and names, affiliations, and email addresses of all authors. Indicate each author who is a full-time student.
2. The author who will be the contact for the Program Committee. Include his/her name, affiliation, paper mail address, daytime and evening phone numbers, email address, and fax number (as applicable).

For more information, please consult the detailed author guidelines at http://www.usenix.org/lisa05/cfp/guidelines.html. **Proposals are due May 10, 2005.**

Training Program

LISA offers state-of-the-art tutorials from top experts in their fields. Topics cover every level from introductory skills to highly advanced. You can choose from over 50 full- and half-day tutorials covering everything from performance tuning, through Linux, Solaris, Windows, Perl, Samba, TCP/IP troubleshooting, security, networking, network services, backups, Sendmail, spam, and legal issues, to professional development.

To provide the best possible tutorial offerings, USENIX continually solicits proposals and ideas for new tutorials. If you are interested in presenting a tutorial or have an idea for a tutorial you would like to see offered, please contact the Training Program Coordinator:

Daniel V. Klein
Tel: +1.412.422.0285
Email: dvk@usenix.org

Invited Talks

An invited talk discusses a topic of general interest to attendees. Unlike a refereed paper, this topic need not be new or unique but should be timely and relevant or perhaps entertaining. An ideal invited talk is approachable and possibly controversial. The material should be understandable by beginners, but the conclusions may be disagreed with by experts. Invited talks should be 45–60 minutes long, and speakers should plan to take 30–45 minutes of questions from the audience.

Invited talk proposals should be accompanied by an abstract describing the content of the talk. You can also propose a panel discussion topic. It is most helpful to us if you suggest potential panelists. Proposals of a business development or marketing nature are not appropriate. Speakers must submit their own proposals; third-party submissions, even if authorized, will be rejected.

Please email your proposal to lisa05it@usenix.org. **Invited talk proposals are due May 10, 2005.** All abstract submissions must be electronic, in ASCII or PDF format only. ASCII format is greatly preferred.

The Guru Is In Sessions

Everyone is invited to bring perplexing technical questions to the experts at LISA's unique The Guru Is In sessions. These informal gatherings are organized around a single technical area or topic. Email suggestions for Guru Is In sessions or your offer to be a Guru to lisa05guru@usenix.org.

Workshops

One-day workshops are hands-on, participatory, interactive sessions where small groups of system administrators have discussions ranging from highly detailed to high-level. The primary goal of workshops is to provide a forum for system administrators to define and develop a given topic in depth. Topics that develop into a community with mailing lists and yearly workshops are especially desirable. Previously successful workshops have focused on configuration management, system administration education, and AFS.

A workshop proposal should include the following information:

- Title
- Objective
- Organizer name(s) and contact information
- Potential attendee profile
- An outline of potential topics

Please email your proposal to lisa05workshops@usenix.org. **Workshop proposals are due May 10, 2005.**

Work-in-Progress Reports

A Work-in-Progress report (WiP) is a very short presentation about work you are currently undertaking. It is a great way to poll the LISA audience for feedback and interest. We are particularly interested in presentations of student work. To schedule your short report, send email to lisa05wips@usenix.org or sign up the first day of the technical sessions.

Suggested Topics for Authors and Speakers

Want to write a refereed paper for LISA or give an invited talk, but don't have an idea for a topic? Here's a list of open questions and research areas that the LISA '05 organizers and a few colleagues created as a good starting place. We're especially interested in papers and talks that address the following topics, but we welcome proposals about all new and interesting work.

- Are all our sites really that different? Are we missing a more standardized methodology? Or are we just all rugged individualists who are causing our own problems (e.g., the account management problem)?
- Can you architect an infrastructure that won't be obsolete in 3+ years?
- Case studies: How do we move from reactive to proactive?
- Case studies: Sometimes we need to see the overall, integrated result, instead of yet another new tool to do the same thing some older tool already does
- Designing machine rooms for the next 3+ years
- DIY-IT (the "IT Garage Trend")
- Dynamic building of coalitions/collaborative environments
- Exploring collaborative tools and "social software"
- Exploring P2P, VoIP, and XML
- How can we design systems that fail-safe by default?
- How do people manage their personal email?
- How do you "manage" your manager?
- How do we train sysadmins to solve problems
- Improving tools for diagnosing problems with systems: Why can't our systems do a better job of explaining what's wrong with them?
- Information sharing: Need to know vs. publish and flag
- Innovative ways to exploit ticket systems
- Intel has suggested it's hit the limit with a 4Ghz processor; how do we deal with this cap?
- It's 2005: Why do we still have computer viruses?
- Managing content and collaborative systems for our customers: We are increasingly being asked to create and manage systems for our users to then expand and "grow." Consider the business use of blogs, wikis, and other systems that can have open-ended growth and change at the hands of "non-professional syadmins" (i.e., users). Do these have special challenges for sysadmins?
- Metrics
- Outsourcing/offshoring system administration: Is it possible?
- People management: More and more "classic sysadmins" are being pushed into first- and second-level management. How do we teach these high-geek people to manage people without growing pointy hair? What should sysadmins be learning to prepare for the day when they wake up managing 3–10 rugged individualist sysadmins (that were likely their peers the day before)?
- Real system administration tools: Why does every sysadmin/site have to roll its own tools?
- Scaling: How do you deal with the next 2x in storage, backup, networking, address space, database, productivity, etc.?
- Scripting languages (stuff around them)
- Selling sysadmin to management: What is your personal ROI? How do we measure it?
- Spam
- The "scaling problem": How do we scale (share) successful tools and processes, content creation and system extensions, and sysadmin experience?
- Tools for understanding information flows in networks and systems
- Virtualization: Benefit or bane?
- What can big sites learn from small sites?
- What have/haven't we learned about picking defaults?
- XML usage for configuration (management)
- Zero administration systems: Why not?

Thanks to Marcus Ranum, Rob Kolstad, and the LISA '05 organizers for contributing to this list.

Contact the Chair

The Program Chair, David Blank-Edelman, is always open to new ideas that might improve the conference. Please email any and all ideas to lisa05chair@usenix.org.

Final Program and Registration Information

Complete program and registration information will be available in September 2005 at the conference Web site, http://www.usenix.org/lisa05. If you would like to receive the latest USENIX conference information, please join our mailing list at http://www.usenix.org/about/mailing.html.

Sponsorship and Exhibit Opportunities

The oldest and largest conference exclusively for system administrators presents an unparalleled marketing and sales opportunity for sponsoring and exhibiting organizations. Your company will gain both mind share and market share as you present your products and services to a prequalified audience which heavily influences the purchasing decisions of your targeted prospects. For more details please contact sales@usenix.org.

Announcement and Call for Papers **USENIX**

Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI '05)

Sponsored by the USENIX Association

http://www.usenix.org/sruti05/cfp

July 7–8, 2005 **Cambridge, Massachusetts, USA**

Important Dates

Submissions due: *March 30, 2005, 11:59 p.m. EST*
Notification of acceptance: *May 3, 2005*
Final papers due: *May 23, 2005*

Conference Organizers

Program Chairs

Dina Katabi, *MIT*
Balachander Krishnamurthy, *AT&T Labs—Research*

Program Committee

Paul Barford, *University of Wisconsin*
Steven M. Bellovin, *Columbia University*
Herve Debar, *France Telecom R&D*
Mark Handley, *University College London*
Doug Maughan, *U.S. Department of Homeland Security*
Chris Morrow, *UUNET*
Vern Paxson, *ICIR/ICSI*
Dawn Song, *Carnegie Mellon University*
Paul Vixie, *ISC*

Steering Committee

Clem Cole, *Ammasso, USENIX Liaison*
Dina Katabi, *MIT*
Balachander Krishnamurthy, *AT&T Labs—Research*

Overview

The Internet is under increasing attack with unwanted traffic in the form of spam, distributed denial of service, virus, worms, etc. Unwanted traffic on the Internet has manifested itself as attacks via many protocols (IP, TCP, DNS, BGP, and HTTP) and popular applications (e.g., email, Web). Recently, attacks combining multiple exploits have become common. Many solutions have been proposed for specific attacks, some of which have had limited success. SRUTI seeks research on the unwanted traffic problem that looks across the protocol stack, examines attack commonalities, and investigates how various solutions interact and whether they can be combined to increase security. Original research, promising ideas, and steps toward practical solutions at all levels are sought. We look for ideas in networking and systems, and insights from other areas such as databases, data mining, and economics. SRUTI aims to bring academic and industrial research communities together with those who face the problems at the operational level. SRUTI '05 will be a one-and-a-half-day event. Each session chair will play the role of a discussant and present a summary of the papers in the session and a state-of-the-art synopsis of the topic. The workshop will be highly interactive, with substantial time devoted to questions and answers. Submissions must contribute to improving the current understanding of unwanted traffic and/or suggestions for reducing it. The Proceedings of the workshop will be published. To ensure a productive workshop environment, attendance will be by invitation and/or acceptance of paper submission.

Topics

Relevant topics include:

- Architectural solutions to the unwanted traffic problem
- Scientific assessment of the spread and danger of the attacks
- Practical countermeasures to various aspects of unwanted traffic (Spam, DoS, worms, etc.)
- Cross-layer solutions and solutions to combination attacks
- Attacks on emerging technologies (e.g., sensors, VOIP, PDAs) and their countermeasures
- Privacy and anonymity
- Intrusion avoidance, detection, and response
- Virus, worms, and other malicious code
- Analysis of protocols and systems vulnerabilities
- Handling errors/misconfigurations that might lead to unwanted traffic
- Attacks on specific distributed systems or network technologies (e.g., P2P, wireless networks)
- Data mining with application to unwanted traffic
- New types of solutions: incentive-based, economic, statistical, collaborative, etc.

Paper Submissions

All submissions must be in English and must include a title and the authors' names and affiliations. Submissions should be no more than six (6) pages long and must be formatted in 2 columns, using 10 point Times Roman type on 12 point leading, in a text block of 6.5" by 9". Papers should be submitted in PDF or Postscript only. Each submission should have a contact author who should provide full contact information (email, phone, fax, mailing address). One author of each accepted paper will be required to present the work at the workshop.

Authors must submit their papers by 11:59 p.m. EST, March 30, 2005. This is a hard deadline—no extensions will be given. Final papers are due on May 23, 2005, to be included in the workshop Proceedings.

The SRUTI workshop, like most conferences and journals, does not allow submissions that are substantially similar to works that have been published or are under review for publication elsewhere. Accepted material may not be published in other conferences or journals for one year from the date of acceptance by USENIX. Papers accompanied by nondisclosure agreement forms will not be read or reviewed. All submissions will be held in confidence prior to publication of the technical program, both as a matter of policy and in accordance with the U.S. Copyright Act of 1976.

How to Submit

Authors are required to submit papers by 11:59 p.m. EST, March 30, 2005. This is a hard deadline—no extensions will be given. All submissions to SRUTI '05 must be electronic, in PDF or PostScript, via a Web form at http://www.usenix.org/sruti05/cfp/.

Authors will be notified of acceptance decisions via email by May 3, 2005. If you do not receive notification by that date, contact the Program Chairs at sruti05chairs@usenix.org.

;login:

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

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

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