;login:

THE USENIX MAGAZINE

October 2003 • volume 28 • number 5

Haiku Contest Winners!

see page 3

inside:

OPINION

Darmohray: Value Added

TECHNOLOGY

Kereliuk: Introducing Voice Over Internet Protocol Technology

SECURITY

Farrow: Musings

SYSADMIN

Haskins: ISPadmin: Managing and Providing ISP Services

Moskowitz: "Eat Your Own Dog Food"

PROGRAMMING

McCluskey: Using C# Properties and Static Members

Flynt: The Tclsh Spot: Client Server Sockets

Turoff: Practical Perl: Using Object Factories

THE BOOKWORM

CONFERENCE REPORTS

USENIX Annual Technical Conference 2003

15th Annual FIRST Conference

USENIX

The Advanced Computing Systems Association

STOP MONITORING YOUR MONITORS.

THE AGE OF AGENTLESS IS HERE.

10 DAY FREE TRIAL. Chances are your agent-based monitoring system has become a monitored system — monitored by you. It's a relationship you just don't have time for. It's time to go agentless. SiteScope from Mercury Interactive constantly monitors your systems, alerting you the instant a problem pops up. And because it's agentless, maintenance aggravations are kept to a minimum. Download a full version of SiteScope for a free 10 day trial.

www.sitescope.com/trial

MERCURY INTERACTIVE

contents

2 MOTD *by Rob Kolstad*

3 Results of the First Haiku Contest

OPINION

4 Value Added *by Tina Darmohray*

TECHNOLOGY

5 Introducing Voice Over Internet Protocol Technology *by Ryan Kereliuk*

SECURITY

10 Musings *by Rik Farrow*

SYSADMIN

14 ISPadmin: Managing and Providing ISP Services *by Robert D. Haskins*

18 "Eat Your Own Dog Food" *by Adam Moskowitz*

PROGRAMMING

20 Using C# Properties and Static Members *by Glen McCluskey*

23 The Tclsh Spot: Client Server Sockets *by Clif Flynt*

29 Practical Perl: Using Object Factories *by Adam Turoff*

BOOK REVIEWS AND HISTORY

33 The Bookworm *by Peter H. Salus*

34 Networking Anniversaries *by Peter H. Salus*

USENIX NEWS

35 2004 USENIX Nominating Committee *by Dan Geer*

36 Membership News *by Tara Mulligan*

CONFERENCE REPORTS

37 USENIX Annual Technical Conference 2003

58 15th Annual FIRST Conference

;login: Vol. 28, #5, October 2003

;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.

$80 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $110 per year.

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

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

Cover: The Portland State Aerospace Society's display in the Exhibit Hall for USENIX '03.

motd

by Rob Kolstad

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

<kolstad@usenix.org>

News from All Over

On the SAGE front, I am excited to announce SAGE's biggest current news: the largest reported barrier to joining SAGE has fallen. SAGE and USENIX dues are now separated! Anyone can join USENIX for $110/year; anyone can join SAGE for $40/year – and there is no longer a requirement for membership in USENIX as a prerequisite for membership in SAGE. If you're in the computer administration world, please urge your friends and associates to join. (See page 36 for details.)

One of the benefits of SAGE membership is access to the results of the annual salary survey. This year's survey was the largest yet (we teamed with SANS and *Sys Admin*, with almost 10,000 participants). Results are now posted on the SAGE Web site. Enjoy!

You probably know that USENIX (this year along with SANS) sponsors the USA Computing Olympiad, the premier high school computing competition (and I am the head coach). Goals including encouragement of careers in computing, in addition to identification and recognition of outstanding programmers.

In 2003, Don Piele (long-time USACO director) hosted the international programming championships (the "International Olympiad on Informatics") at his institutional home, the University of Wisconsin–Parkside. Fully 75 countries competed (about 300 competitors) in the week-long event held August 16th-23rd which included 10 hours of grueling programming competition.

For the first time ever, the USA team tied for highest honors, with two gold medals (for juniors Alex Schwendner and Tiankai Liu) and two silver medals (for junior Anders Kaseorg and MIT-bound Timothy Abbott). Being the host country, a second team also competed for honor and earned scores that otherwise would have earned medals. Only one senior is graduating this year, so we've got great chances for next year, as well. Best of all, the competition was run flawlessly for the first time, with no glitches or major protests. Congratulations to the competitors, coaches, and organizers.

As I type this, the master copies of the LISA Conference proceedings are printed and shortly to be sent to the publisher. The subjects range across the spectrum from Alva Couch's new theory work to Josh Simon's and Liza Weissler's super practical document repository implementation.

A new "lunch with the speakers" program will widen access to presenters, who will appear at informal tables in the courtyard shortly after their talks for lunch (or, in the case of afternoon presenters, late-afternoon drinks). Yet another reason why LISA is the premier conference for administrators.

Several topics are appearing on my radar this month, and space won't permit me to write 650 words about all of them. They are interesting enough, though, that I hope you'll think about them and let me know if you hear report-worthy developments. They are presented here in no particular order.

Some Predict the Death of the Internet

I'm hearing more pundits (including some traditionally associated with the USENIX community) bemoaning current Internet problems. Of course, the recent worms/viruses have done nothing to help, but spam and the general noise level (including pop-up ads for browsers) has prompted ever more to suggest that the usefulness of email and the Web is declining. Some suggest that solutions involving balkanization (creation of smaller [sub-]nets that serve specific [potentially distributed] communities) is the only answer. I imagine that will happen anyway, but more as an addition than as a partition.

Some Predict Quicker Rise of Linux

Pundits, *Information Week*, and of course the vocal evangelists are saying that Linux will rise to become a/the dominant operating system. Larger corporations (who really do move most of the money in the IT world) are now in the "when" phase of Linux deployments rather than the "if" phase, according to *Information Week* magazine polls and others. Why mention this now? Because the security issues are really pushing management to understand what's going on in the machine rooms, performance and usability are now seen as assets, and the SCO lawsuit is providing publicity and visibility to Linux. Regrettably, the rise of Linux means the death-knell for the BSD line.

Mixed Predictions on Future Job Markets

While outsourcing comments led all others in the free-format section of the salary survey, many on the Net also voice concern about future careers in IT, administration, software, etc. New criticisms are appearing that academic education is diverging from the needs of industry. The worries include: lack of curriculum relevance means college graduates are not as employable, industry will have to step in to make its own program (which will surely be more focused on the near term), enrollments in traditional programs will drop even more (they're already way down from highs during the dot-com boom), and foreign outsourcing will drive anyone still interested into other careers, anyway.

More next time.

Results of the First Haiku Contest

Two months ago, I asked people to send me haiku about the Internet. What a fabulous set of entries I received! I culled them down to some of the best, shown below. Thanks to all who wrote; the topic of the next contest is at the end of this article.

Matthew W. Hurlburt's first entry worked in a sort of anthropomorphism, raising the specter of digital isolation:

Alone in my room
Digitally connected
To others like me

Matthew also sent a more traditional haiku, that included both technology and a season:

Like a Spring rainstorm
The server stops and restarts
Over and over

Steve C. Johnson submitted several snippets; here are my favorites:

Innocent action
violates segmentation
it's a null pointer

For systems admin
Professional adventure
come visit LISA

UNIX eternal
From PDP-11
To my PDA

Edward Chrzanowski sent in an interesting set of haiku where changing spacing or single letters makes for new haiku with new meanings. Consider changing "infinite" to "in finite" or changing "wander" to "wonder":

Stateless I wander
My passage HTTP
Infinite roaming

My favorite of Tobias J. Kreidl's submissions ended with a twist, a sort of apocalyptic tone:

Slipping though a pipe
Bits propagate through routers
Filling us with fright

One of John Sellens's entries sported a similar theme:

Contemplating bits
Intrusion detection works
Machine is now owned

My favorite (though it's really hard to choose!) was Heather Fox's more traditional entry, with its digital, emotional, and outdoor themes:

Shooting stars of bytes
breathing ether in and out
this world has no sun

Heather wins the first haiku author polo shirt.

This was great fun!

Let's try for another topic. Compose a haiku that reveals the joys or frustrations that surround the process of developing scripts or programs. Send it to *haiku@usenix.org.*

Haiku are little three line poems with five syllables in the first line, seven in the second line, and five more in the last line. Syllabic stresses (accents) are not important. The brief haiku captures a thought, a moment, a scene, or a vision. The best haiku amplify insight or even cause an "Aha!" or epiphany. Many haiku traditionally describe the seasons of the year.

;login:

EDITORIAL STAFF

EDITOR:
Rob Kolstad *kolstad@usenix.org*

CONTRIBUTING EDITOR:
Tina Darmohray *tmd@usenix.org*

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

COPY EDITOR:
Steve Gilmartin

PROOFREADER
jel *jel@usenix.org*

TYPESETTER:
Festina Lente

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

opinion

Value Added

by Tina Darmohray

Tina Darmohray, contributing editor of *;login:*, is a computer security and networking consultant. She was a founding member of SAGE. She is currently a Director of USENIX.

tmd@usenix.org

I was fortunate to land in a soup-to-nuts project the first job I had out of college. I had everything to learn and there was at least one of everyone to learn from. We had scientists, kernel coders, application programmers, database programmers and administrators, documentation specialists, system administrators, networking and telecom jockeys, and so on. Despite all the different flavors of computer professionals on the project, I noticed one universal theme: Those who understood the whole picture were the most valuable.

Since I had everything to learn and didn't have a particular specialty yet, I benefited from every person on the project. While there, I did some programming, project management, technical training, documentation, and database and system administration. I was like a sponge, and there was free-flowing knowledge running steadily throughout the hallways and conference rooms of that project. I did my best to come up to speed on my own, but when I had questions there were always folks to go to for answers. There was so much to learn and, luckily, so many really good people willing to let me learn from them.

As I came out from under my initial input overload, I began to notice that I wasn't the only staffer who sought help from colleagues. In fact, there was information exchange going on for everyone, regardless of their experience or seniority. After a while I noticed a trend, though: There were some folks who were the top question-answerers, and it was almost as if an unofficial org chart was reflecting it. It was these folks whom the experienced staff went to when they had questions. And it was these same people who had the last word in meetings and to whom everyone deferred when the strategic decisions were being made.

Since their authority was clear, though not official or necessarily aligning with a title of "manager," I began to assess what it was they had in common. Were they the scientists on the project? The kernel programmers? Soon I realized that it wasn't their profession or title which they had in common but their body of knowledge. In every case these local sages were the people who knew more than just their area of specialty. For the programmers, that meant they knew the kernel as well as the applications and networking. For the network folks, it was the guy who could talk with the kernel programmers. For the sysadmins, it was the one who could administer the network applications as well as the local machine, and for the database folks it was the person who understood the OS and its administration, too.

In every case, these people were able to bridge the gap to their peers. On this particular project, which drew on so many different disciplines, those who could do so were key. But I've observed this type of employee is always key, no matter where they work. It's like they are translators in the tower of Babel, the hub of a communications wheel. For their ability to do and understand more than one thing, they are more valuable and more utilized than their peers.

Thinking back 20 years, I realize that additional value hasn't changed. The people who understand the way things work, top to bottom, are still the most valuable and influential folks on any project. In this economy, being that person may mean you keep your job or get the next one. Take every opportunity to broaden your body of knowledge; not only will you bring added value to your project, you'll also add value to your resume.

introducing voice over Internet protocol technology

by Ryan Kereliuk
Ryan Kereliuk is a contract developer focusing on systems and network software ranging from device drivers to protocol stacks. He holds a master's degree in CS from the University of Alberta.

ryker@ryker.org

Overly optimistic marketing during the so-called communications revolution has given voice over Internet protocol (VoIP) technology the stigma of being the next big thing that never materializes. I'll risk the same mistake by suggesting that VoIP is emerging as a viable Internet application.

Key VoIP Drivers

Broadly speaking, three factors will motivate the adoption of VoIP:

- Reduced ownership and operational costs
- Simplification
- A roadmap for building next-generation services

Operational cost is paramount. Long distance charges are the first and most obvious expense. Skeptics suggest that VoIP has missed its window because of fierce competition in the long distance market, but the market continues to grow, so the window remains open. Furthermore, even a few cents a minute is far more expensive than utilizing an otherwise unused – and already paid for – resource.

Personnel costs also add up. Most companies large enough to have their own private branch exchange (PBX) are staffed with a telecom group. Since VoIP is premised on open, interoperable standards, telephone services become an application more akin to running an HTTP server than a traditional phone system. One is able, then, to leverage the competencies of an IT networking group and invest resources there, which is attractive given the versatility of those staff.

Security is a feature that one gets "for free" with VoIP. VoIP is secured in the same way as other Internet services: by minimizing attack vectors, using strong authentication, and protecting important servers with a firewall. On the management front, IP phones can often be configured using DHCP and TFTP, for instance, which exploits services that usually exist already.

Simplification might save money directly but is a benefit in its own right. VoIP converges voice and networking, reducing the number of service providers a company must deal with. (Of course, I'm not predicting the death of the public switched telephone network or PSTN in the short or medium term.) On one hand, ISPs offer straightforward billing plans for bit pipes. On the other, VoIP empowers people to take control of the server infrastructure that telcos use to extract complex fees for every move/add/change operation.

The most important simplification, though, concerns the phones themselves. If VoIP is really just another Internet application, then are phones even required? So-called hard phones are available to recreate the experience of using a regular phone, but soft clients, ordinary PC applications that run on desktop computers, are making inroads. In time soft clients will dominate.

VoIP will offer new features over legacy phone systems, namely rich media and convergent integration. Rich media rejects the assumption that a communications channel spans only one medium (e.g., audio). Instead, voice, video, and text can be shared simultaneously. VoIP protocol engineers, particularly those with the Internet Engi-

Estimates suggest that VoIP can save 30% over conventional telecom rollouts after accounting for long distance charges, personnel, servers, and phones.

neering Task Force (IETF), have shown tremendous commitment to generality, so today's protocols are equipped to help people communicate in whichever ways serve them best.

Convergent integration – unified messaging – suggests that voice mail and faxes should, like email, be accessible using any device connected anywhere on the Internet. This unification might even extend to integrating VoIP software into customer relationship management suites or e-commerce applications.

The case for VoIP is closely tied to cost as well as potential new services. Estimates suggest that VoIP can save 30% over conventional telecom rollouts after accounting for long distance charges, personnel, servers, and phones.

Thinking About Deployment

A VoIP project, not surprisingly, begins with a requirements analysis that informs the trade-offs faced at every stage of the decision-making process. Key areas for consideration include cost, network requirements, protocol selection, client and server hardware and software, impact on existing infrastructure, and migration paths.

Regardless of the underlying technology, certain questions govern subsequent choices and, in the end, serve as evaluation criteria for any deployment. Three that deserve consideration are:

- System utilization
- Interoperability
- Quality

Utilization refers to the capacity of a given system and is related to the numbers of active and potential users: How many VoIP endpoints will be connected, and what percentage of those will be active at any one time? Interoperability is concerned with which users can connect with one another: Is the system meant for internal use only or for use with external parties? In the latter case, are external parties reached by using some (possibly different) VoIP protocol or by using a gateway to the PSTN? Finally, unlike legacy phone systems, VoIP offers the opportunity to trade quality for other benefits. What sort of quality do users expect in terms of system availability and media fidelity?

Call quality is governed by the codec in use, assuming the network transport is able to keep up. The term *codec* is familiarly expanded to coder/decoder but, these days, compressor/decompresser is an equally valid meaning. Different media codecs require different network resources but also provide different levels of quality. The G.711 codec, for example, provides quality equal to conventional telephone systems at a rate of 64Kbps. G.729A, by contrast, needs 8Kbps of bandwidth but sacrifices quality. There is no substitute for making test calls with different codecs, but my subjective impression is that G.729A offers a quality markedly better than cellular telephone service. Choosing a codec or building a model of codec usage is of great importance to resource planning. Simplistically, the codec bandwidth can be multiplied by the number of active users or voice paths to compute bandwidth needs.

A bandwidth number in hand, one can begin shopping for network service. Often forgotten, though, is that bandwidth is just one measure of a network. Latency, jitter, quality of service (QoS), and availability are other important considerations. VoIP systems are particularly sensitive to latency, which distorts the flow of conversation, because changes in speaking order are politely signaled by silence. High latency has the effect of making multiple parties think the channel is open to new speakers – which of course results in a collision. A rough rule of thumb is that end-to-end latency should not exceed 250 milliseconds, but this number should be minimized. Jitter is a measure of the difference in inter-packet latency between packets leaving the sender and arriving at the receiver. Introduced by network elements, jitter can cause increased packet loss or perceptible delays, so its minimization is very desirable.

Making guarantees about bandwidth, latency, and jitter – QoS in networking parlance – is a hard problem that is not adequately solved in contemporary networks. These days, poor man's quality of service is achieved by overprovisioning, a very basic but effective technique. In summary, if one is outfitting a new location with network service or is changing service providers, it is a good time to ask harder questions than one might have in the past. Learn about a provider's reliability by interviewing existing customers, get comfortable with their problem-tracking and resolution procedures, and sign a service level agreement (SLA) that captures the important requirements.

Media codecs have direct implications for bandwidth. By contrast, calls are set up and torn down with a signaling protocol that is comparatively lightweight. When it comes to signaling, one is able to choose from a wide selection: Megaco, H.323, SKINNY, MGCP, SIP, etc. (to name a few); a comparative examination of these different protocols is beyond the scope of this article. The only protocol I work with today is the IETF's Session Initiation Protocol (SIP), which I strongly recommend to anyone implementing VoIP.

The signaling protocol one selects will obviously affect choices to do with client and server software and, to a lesser extent, client and server hardware. For SIP, many server choices are available, including high-quality open source and commercial packages. In terms of hardware, a good rule of thumb is that the machine used for an organization's corporate HTTP server is comparable to the machine needed to run SIP services.

On the client side, choosing between soft clients and hard clients is a key decision. Soft clients tend to be less expensive (or free), offer more features, and integrate more organically with existing desktop software. The downside, if there is one, is a learning curve similar to deploying any new application. Hard phones, predictably, offer a user experience that in most cases is identical or slightly improved over regular telephone handsets.

Regardless of the particular protocol choice, the infrastructure associated with deploying VoIP will be impacted. Hard phones, for example, don't reuse the resources allocated for existing PC desktops. When deploying hard phones, additional switch ports are required for each endpoint, more cabling may be needed, and routers are a factor if additional subnets are required.

Nodes (whether hard phones or desktops) configured with RFC 1918 private IP addresses will have issues communicating with people beyond the nearest network address translation (NAT) boundary. SIP, for example, includes routing information inside IP packet payloads, which means that vanilla translation systems do not work. Like FTP, it will be some time before NAT-enabled firewall devices are smart enough to rewrite these packets with appropriate translated address information. In the meantime, options include using public IP addresses for SIP endpoints or using a SIP-specific application-level gateway that is able to reconcile addresses inside SIP packets.

Lastly, any VoIP implementation needs to mesh with existing security policies and infrastructure. This might mean adjustments to deployed security systems such as fire-

Any VoIP implementation needs to mesh with existing security policies and infrastructure.

walls and RADIUS servers. VoIP security is particularly important, especially considering that authorized users can often access expensive resources such as PSTN lines through a gateway.

Finally, finding a migration path will be of key importance if one already has a significant investment in legacy phone technology or cares about reaching PSTN users. PBX users are best advised to contact their PBX vendor, as some vendors have VoIP options in the form of line cards for existing equipment. If PSTN connectivity is needed for a VoIP installation, a variety of choices are available in a variety of sizes. Running a T1 line to a VoIP-enabled router, for example, allows 24 simultaneous calls to be gated to the PSTN. On a smaller scale, some routers can take a module implementing a foreign exchange office (FXO) interface, which connects one or two lines to a telco. In short, telephony companies are sensitive to migration issues, and solutions are generally available.

An Example Implementation

Building out a VoIP network is not as complex as it might seem. In this section I will give a high-level description of one setup I've used that addresses some different requirements. This scenario describes the VoIP solution for a multi-office distributed company.

In this deployment, remote offices are connected to the Internet using business-grade DSL lines. When using this class of network connection, one generally doesn't have the influence necessary to negotiate a favorable service level agreement, but the good news is that these links are more than adequate for serving small satellite offices. The central corporate phone system is served by a VoIP-dedicated one-megabit link with an SLA guaranteeing latencies of 70 milliseconds or less to other predefined points on the Internet. This is a connection capable of serving 12 calls with G.711 or 42 calls with G.729A, though both codecs are used in practice.

This example uses the SIP signaling protocol exclusively, with a hodgepodge of different servers and clients. In terms of servers, for example, remote offices utilize open source SIP proxies, including VOCAL and SIP Express Router (SER), while the headquarters uses a commercial SIP server that supports unified messaging functions. The clients deployed vary widely, because the organization is a software development firm in which developers are permitted to use the client of their choice. Typically, however, desks are equipped with hard phones such as Cisco 7960 handsets, while roaming users use soft clients such as Microsoft's Messenger product.

The reality is that VoIP is not currently sufficient to reach all potential business partners and customers. In this deployment, then, PSTN connectivity is supplied using Cisco routers. One satellite office uses a Cisco 2600 router equipped with an FXO module to connect two lines to the PSTN for local dialing. Headquarters, however, uses a Cisco 3600 router with a T1 interface card to provide PSTN direct dial numbers into 24 different VoIP endpoints. By terminating these latter PSTN lines at SIP phones, some of them remote, the organization achieves the effect of virtualizing its geographically distributed operations at the head office. Both PSTN compatibility systems work very well.

On Timing

There is no question that VoIP represents a major paradigm shift. The legacy telephony model is very strong and thus will not be unseated easily – and change will not happen all at once even in an environment of enthusiastic adoption. So when, then?

Before I say when, let me describe how. The value of a communications medium correlates with the number of users reachable using that medium, so the adoption rate will accelerate due to what economists call a network effect. This is developing in a couple of ways.

First, trends in the service provider sector augur well for the emergence of VoIP. We're presently seeing major telecommunication providers size up the next generation of services. Looking to the competitive long distance market, we already see more agile providers using IP links to traffic international PSTN calls, thus gaining a competitive edge.

There is a new breed of company, those offering local and long distance telephone service using VoIP technology. Vonage, for instance, offers flat-rate local and long distance calling in the United States, with the option of choosing a phone number from several American area codes. Early entrants like Vonage are aiming to capture business and consumer customers who aren't in a position to manage their own server-side infrastructure or PSTN interconnectivity. This parallels managed Web services, which coincided with the growth of that technology.

Second, consumers, too, are seeing new reasons for VoIP, including the sort of managed services just described. Other reasons include network access and software availability. As anecdotal evidence, I've been broadband connected for over five years, and among my friends, family, and coworkers this is universal (although Canada is a bit ahead of the curve on this technology). And, now, inexpensive VoIP software is widely available to leverage such connections. By the time this article is printed, Microsoft, the desktop juggernaut, will have released the latest version of their Windows XP Messenger program, which includes a complete SIP client. This enhancement, to be released in version five at Microsoft's Windows Update site, means that over 17 million PCs are potential VoIP nodes using the SIP protocol.

Remember the network effect? Within 36 months it will be possible to satisfy a healthy percentage of one's calling needs, both personal and professional, using VoIP.

Conclusion

It has been the intent of this article to raise consciousness about VoIP, a technology that has been rightly considered imaginary. I use VoIP systems every day for my telephone needs, so, to me, they hold great promise. I'm convinced that VoIP is, at once, both a solution for next-generation communications and a challenge for IT departments, so it would be wise to keep VoIP in mind. The good news is that people comfortable with networking and Internet services will catch on to VoIP very quickly.

In a future article, I will focus on the Session Initiation Protocol for VoIP. In that article, I'll look at some of the lower-level nuts-and-bolts related to implementing SIP services. I'll describe the different actors in a SIP system, with examples drawing on open source SIP applications. SIP is just an application protocol that runs on the Internet, so, as developers, we can contemplate writing new and interesting phone services without investing huge amounts of time learning proprietary protocols. To that end, we'll take a tour of one open source application built on an open source SIP stack.

SELECTED POINTERS

http://www.voip-calculator.com/ – An online script to calculate network requirements based on anticipated utilization.

http://www.vovida.org/ – An open source community site that offers a variety of VoIP software including the VOCAL SIP proxy.

h*ttp://www.iptel.org/* – Makers of the SIP Express Router proxy server and providers of personal SIP hosting services.

http://pulver.com/fwd/ – A free SIP service provider that currently connects over 40,000 users.

http://www.microsoft.com/windowsxp/windowsmessenger/ – Watch here for the new Windows XP Messenger or download the old Messenger 4.7, which is also SIP compatible.

h*ttp://www.vonage.com/* – A service provider offering SIP services with interconnection to the PSTN.

http://www.resiprocate.org/ – An open source SIP stack optimized for speed.

http://www.asteriskpbx.org/ – An open source, multi-protocol PBX suite for Linux.

http://www.xten.com/ – Xten offers the lite version of their SIP soft phone as a free download.

http://www.sipphone.com/ – A SIP provider that sells hard phones for use with its free service.

musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V*.

rik@spirit.com

In my last column, I waxed enthusiastic about how superior Windows was when it came to the ease of writing keystroke sniffers. While there are hundreds more keystroke sniffers for Windows, I have since learned of a new one that runs on Linux (2.2 and 2.4 kernels), Solaris, and OpenBSD.

I was talking to Ed Balas, after being grabbed by Lance Spitzer and dragged over to a table staffed by the Honeynet Project at the 2003 BlackHat conference in Las Vegas. Ed explained to me that he was a "generalist, not a specialist," and had just finished writing, then porting, a kernel module that is designed for use in GenII Honeynet, which he is writing about for the December 2003 annual security issue of *;login:*.

The original honeynets relied on packet sniffing, and that can be a problem when the first thing the attacker does is download and install a version of SSH. Now, everything that gets downloaded can travel across an encrypted link. And attackers have started to encrypt their tools on UNIX systems, something that has been done for a while with Windows trojans and viruses, so static analysis becomes much more difficult. But if the honeypot can capture all keystrokes, the potential for useful research becomes much greater for the honeypot owner.

Sebec2 (see *http://project.honeynet.org/papers/honeynet/tools/index.html*) hooks into the read entry in the kernel's system call table, so that *anything* that a program reads, whether it be a file or a command line, gets captured. That means that any encryption keys or passwords that an attacker uses will be recorded. Sebec2 then creates its own IP packets and writes them directly to the NIC device driver. Most applications that create their own packets (e.g., hping) write to a raw socket, and that makes it possible for snort or tcpdump to capture the packet before it gets written to the device driver. But Sebec2 prevents the packets from being detected by any application-level process – in fact, by anything other that a kernel module like Sebec2 itself. The server end of Sebec2 still sniffs the network to collect the packets.

As Ed described his new tool, I couldn't help but think that this would make a great rootkit. Sebec2, unlike xkey (see my August 2003 *Musings*), also reports the application that requested the `read()`, so an eavesdropper gets a much clearer picture than with xkey. And Sebec is listed as a rootkit at chkrootkit.org. Like many other security tools, the potential is there for both good and dark uses.

I also heard about a really interesting study done by Cisco employees. They decided to take a closer look at the myths surrounding BGP vulnerabilities. When someone claims to be able to "bring down the Internet in a half hour," they would be referring to flaws either in the routers or in the core routing infrastructure, and that means BGP. Matthew Franz and Sean Convery first presented their results (publicly) at NANOG and subsequently at the BlackHat conference. You can read their presentation at *http://www.nanog.org/mtg-0306/pdf/franz.pdf*.

I wrote about the potential for "bad things happening" to BGP back in April of 2002, and nothing has changed since then. True, there are now two proposals for adding security to BGP updates, when before there was only one. But S-BGP (Secure-BGP) requires a PKI, the signing of routes by the originating router, and the checking of signatures by the router receiving the update: *http://www.ietf.org/internet-drafts/draft-clynn-s-bgp-protocol-01.txt*.

I heard complaints about this concept from many people, who pointed out that handling public key encryption was not something router CPUs could manage while still getting useful work done. The S-BGP proponents do not consider this an issue, but if you do use BGP, and S-BGP becomes mandatory, plan on updating the core of your router so it will include the extra processing power necessary. The router company version is called SoBGP, or Secure Origin BGP, and differs in that it uses a decentralized certificate management system and relies on RADIUS servers (or something that does *not* involve the router's CPU) to handle public key encryption: *http://www.ietf.org/internet-drafts/draft-ng-sobgp-bgp-extensions-01.txt*.

Misconfigured routers are . . . a much bigger issue at present than problems with BGP.

Convery and Franz decided to take a hard look at the FUD surrounding BGP by creating attack trees, then trying out each branch to see if any of these attacks was actually something to worry about. Recall, if you would, that BGP neighbors, or peers, keep a TCP connection going at all times. Once a minute, they exchange keep-alive messages, or they might send routing updates to their neighbors. If this TCP connection (to port 179) goes down, each router strives to recreate it quickly while also sending out updates reporting that the routes they know of that travel through the neighboring router are all invalid. That information percolates across the Internet, forcing any router with a complete set of BGP routes to recalculate its routing tables. In Convery and Franz's attack tree, this leads to two branches: disrupt the TCP connection or insert phony routes (you can read the papers or presentation for much more detail).

So they tried disrupting the TCP connections between seven different products/implementations that all support BGP. And failed. They also tried hijacking BGP connections. BGP neighbors can use MD5 authentication that gets included as a TCP option, and if this is used, hijacking means guessing the shared secret. But hijacking also means being able to sniff communication between BGP neighbors, which are often connected using a link dedicated to that purpose. The short answer is that hijacking a connection could be done if MD5 authentication is not used, and the connection can be sniffed.

They also attempted inserting an update when MD5 was not used, and this resulted in an ACK storm (since the receiving system has sent an acknowledgment for a packet that the sending system has never heard of), with the eventual result that the connection gets reset after five minutes. But during that five minutes, the phony routes spread through the Internet (unless blocked by BGP route filtering, but that's another story).

They tried fuzz testing using BGP messages in attempts to crash BGP; four flaws were found in 1200 attempts, and three of those worked only if received from a valid peer and/or valid AS. The fuzz testing included sending messages that varied from totally random data to up to eight valid fields followed by random data. So the implementations appeared to be fairly robust.

The most interesting test that Convery and Franz carried out was to use traceroute to 120,000 destinations, one for each CIDR block in the Internet, in an attempt to map out all the routers that might be BGP speakers. With a list of 115,466 IP addresses ready, they then tried SYN probes against each address, to ports 22, 23, 80, and 179 (SSH, Telnet, HTTP, and BGP). In theory, any access to administrative ports by "outsiders" should be filtered or ignored. In practice, 14.5% of all routers responded to at least one of the three administrative ports, SSH, Telnet, or HTTP.

They concluded, based on other tests not mentioned here, that the best way to disrupt the Internet is to take over one or more routers and have them distribute misleading routes. If a trusted router gets compromised, signed updates (if these are eventually approved) would still be trusted. Misconfigured routers are, they concluded, a much bigger issue at present than problems with BGP.

Each vulnerability stretches way back in time, to much earlier versions, yet is still present in the Windows Server 2003.

July 2003 also brought with it announcements of several new buffer overflow vulnerabilities in Windows. One announcement affected all versions of Windows from 95 up to Server 2003 (CA-2003-14 or MS03-023) and the module that handles Rich Text within HTML. The other involved only NT through Windows Server 2003, and requires access to port 135 TCP or UDP (CA-2003-16/19 or MS03-026). Windows Server 2003 is supposed to be the most secure version of Windows, so the existence of these vulnerabilities was considered an embarrassment for Microsoft. But I noticed something other than simple embarrassment in these vulnerabilities.

Each vulnerability stretches way back in time, to much earlier versions, yet is still present in the Windows Server 2003. What that tells me is that Windows Server 2003 is still using code left over from the earliest days of Windows with a TCP/IP stack (the RTF-HTML bug) or from the earliest days of NT. I have always thought one of the biggest issues with Windows systems is the amount of legacy code each version contains, crucial to maintaining backwards compatibility. Microsoft built all of Server 2003 with stack canaries, similar to the StackGuard canaries for Linux, but these servers are still vulnerable.

One reason is that stack canaries can only guard against stack overflows, not heap overflows. An even better reason is that some mistakes in code are so subtle that code analysis by programmers and by special tools will not catch them. The heap overflow found in Sendmail in March of 2003 was a good example, as understanding the source code meant single-stepping through it, a laborious process.

I got a chance to read most of *Secure Coding*, a new O'Reilly book by Mark Graff and Ken van Wyk, both currently with CERT. Secure Coding is about principles and practices, and it provides little in terms of code examples (and most examples are the mistakes, not solutions). Their goal is not to write a programming guide, since other people have done that well, but instead to share their years of experience with vulnerabilities. Problems offered go beyond programming mistakes, such as the all-too-frequent buffer overflows, to design and operational mistakes.

They do include some great examples of failures and of good design. I liked their explanation of how one could log in to some rlogin servers using the login name -froot and get a root prompt with no password (a problem that occurred because rlogind calls the login program without checking the provided login name). Or how /etc/passwd file fragments started showing up in Solaris tar files in a new release. It turned out that some heap memory had just been freed, and then reallocated as the buffer used when writing out tape blocks, and that memory had previously been used in password file reading functions (getpwent()). Both of these issues are problems in composition – that is, neither occurred on its own, but required a combination or sequence of events for the bug to assert itself. In the case of tar, some code had been removed, moving the freeing of the memory used by getpwent() closer to the malloc() used for tar's block buffers. Who would have ever guessed (much less figured this out) before it was discovered the hard way.

Secure Coding is a relatively short book (177 pages), but not an easy one. It really helped me to understand why I was never a great programmer – my programming style was more prototyping than design. Prototyping was loads of fun, as I would occasionally knock together example programs thousands of lines long in a day when I was teaching people how to use a graphics library. I was proud that I could get such a monstrosity to compile, link, and actually demonstrate potential solutions to the problems the client had. But the real problem arose after I left, when the client decided not to toss those thousands of lines of disorganized code, but instead to use it as a starting point for their application. Note that what I was doing was gluing together many short example applications, not writing fresh (but unorganized) code.

Graff and van Wyk provide methodologies and questions that you can use in architecture, design, implementation, and operation. Having read their book, I finally recognized that secure coding cannot be taught as an isolated subject but needs to be woven into every area of software instruction. I certainly recommend this book if you have anything to do with software (beyond just running off-the-shelf apps), as security is important and not something that can be taken lightly (or worse, something that you can just expect will happen organically, without careful thought).

I finally recognized that secure coding cannot be taught as an isolated subject but needs to be woven into every area of software instruction.

As I finish writing this column, I am also getting ready to leave for the USENIX Security Symposium in Washington, DC, something I have been looking forward to for months. BlackHat is interesting, and some research does show up there, such as the Cisco presentation. BlackHat led off this year with a presentation by David Litchfield, who, in his pre-conference notes, planned on explaining the differences between exploits in Linux and Windows using stack-based buffer overflows in Oracle 9i's XDB as the example. Instead, he chose to talk about the much more current issue of the overflows in Windows RPC.

Litchfield certainly knows his stuff, that being the internals of Windows and Linux executables and libraries. Although his presentation was disjointed, he did manage to take the audience through a code debugging session of MS RPC, showing where the problem occurs and how it can be exploited. He could not demonstrate an exploit for Server 2003, which LSD claims to have written but not released (yet). But he did have parts of his audience absolutely mesmerized.

A group of Microsoft employees were sitting in the front of the room, furiously taking notes. Litchfield described how the stack overflow protection works in Server 2003 and suggested methods for evading it, something LSD appears to have done.

This vulnerability in MS server products has great potential for worm writers, as the MS-Blaster worm and its variants have shown. Microsoft, having been slammed by Slammer, had adopted a much less tolerant view of people who fail to patch their systems. Everyone within Microsoft was given a deadline to install the patch for MS03-026. If they failed to do so by the deadline, their network connection would be disabled. I asked a friendly Microsoft employee exactly what that meant, and he told me that internal security was aggressively scanning for the vulnerability, and would turn off the switch port of anyone not patched by the deadline. Also, anyone logging in remotely would also be tested before their authentication could be completed. Wise moves. Apparently, Microsoft's customers did not take the issue seriously enough, as MS-Blaster has been moderately successful, although paling in comparison with the SoBig.F virus. When will they ever learn?

ISPadmin

Managing and Providing ISP Services

by Robert Haskins

Robert D. Haskins is an independent consultant specializing in the Internet Service Provider (ISP) industry.

rhaskins@usenix.org

Introduction

In this edition of ISPadmin, I look at ways service providers manage and provide mail, Web, dialup, and other types of services to their customers. If I were bringing up any service provider type of business from scratch today, ISPMan (or a similar, usually custom-built package) would be at the core, along with an appropriate billing system.

Background

In most smaller "legacy" ISPs, shell scripts are used to provision services ordered by the customer. These scripts are custom written and driven by the "legacy" billing system. Such provisioning applications tend to take on a life of their own, causing a major maintenance headache for the ISP. For example, each time the provider offers a new type or classification of service, the custom provisioning system must be modified to include this new service. Also, when a new system is added, the provisioning system must be modified to include this new system. When the business model changes, there is a high likelihood that the provisioning system will also need to be changed.

There are many operational issues as well. For example, non-LDAP-enabled systems (smaller providers who use stand-alone files for enabling dialup (RADIUS) accounts) experience a short amount of downtime when the RADIUS service is reloaded after updating the user list. Also, there is always a possibility the service won't start up correctly when the associated RADIUS daemon is reloaded after adding a user to the stand-alone file.

Most larger service provider operations have resorted to writing custom scripts for this purpose. However, for a smaller provider (and even for larger ones just starting out), an application like ISPMan is a better option. While somewhat constrained by the inherent design in any application, it is written in Perl so the provider can add functionality if need be.

Open Source Solutions

Several open source systems manage and provide ISP services. Vishwakarma, perhaps the oldest service provider management system, doesn't appear to have been modified in the past three years. It has a lot of the functionality that an ISP would require, such as virtual email, Web-hosting management, and reseller support, and it has an LDAP directory to tie everything together. However, it appears to have several pieces missing. The biggest holes would be lack of a distributed model for back-end services, in addition to quota support. However, it is certainly worth a look. It is released under the GNU GPL.

Another system in the same vein is a package called "vhost" (virtual host), which is also released under the GNU GPL. It has much of the same functionality as ISPMan, with support for virtual email and Web sites. However, its biggest drawback is the lack of an LDAP directory, which would allow a distributed model for enabling back-end services. Hopefully, this is an area of focus for future development. Imagine how the growing ISP would have to throw out the vhost system when they ran out of spare cycles on their existing hardware! It would be much easier, and more cost efficient, to simply add machines to the cluster in order to add capacity.

Commercial Solutions

There are a number of commercial products in this space. Most companies have products in the "virtual server" space as well, which allows a service provider to make (and sell) multiple virtual servers out of one "real" server. (The best known open source virtual server project is vserver.) Many of these commercial products are based upon open source components for their back-end services. Some commercial solutions even manage Microsoft-based products such as Exchange and Windows Server 2000. (As an aside, if you are looking for an open source Exchange replacement, the SAGE members list had a great discussion on the topic back in June 2003.)

The obvious benefit of a commercial solution is the improved support one should get utilizing such a product. Given the level of support available in open source applications such as ISPMan, such differences are decreasing by the day. However, commercial implementations might be useful in certain environments and are certainly worth investigating.

What Is ISPMan?

ISPMan consists of a collection of back-end services (all utilizing an LDAP directory for user information) and a set of custom Perl programs to manage those services. Like the other open source ISP managers, it is has been released under the GNU GPL. The major applications ISPMan uses to provide services to end users include:

- Postfix (mail exchange)
- Cyrus IMAP (POP3/IMAP subscriber access)
- Apache 1.x (Web services)
- BIND 8/9 (name services)
- PureFTP (FTP access)
- OpenLDAP 2.x (directory services)
- FreeRADIUS (RADIUS services)
- Horde IMP (Webmail)

The system is scalable, and redundant with appropriate additional hardware. It contains a Web interface for management/provisioning, as well as a command line interface for provisioning accounts via automated mechanisms (e.g., through the service providers billing system) and for other tasks. Any function that can be performed via the Web interface is available via the command line interface on any node in the cluster! It is worthwhile noting that the developers of ISPMan plan on implementing a Simple Object Access Protocol (SOAP) interface for easier integration into the billing system, and other required interfaces as well.

ISPMan has a very nice "reseller" feature, where the ability to add/change/delete services can be delegated to others (e.g., a wholesaler setting up a reseller to sell dialup accounts, Web hosting, etc.). Also, this interface could be used for a large enterprise, where the central IT department gave access to people in each division to add email or dialup accounts, perhaps even under their own subdomains (or completely different domains). This is discussed in "How an Enterprise Might Use ISPMan," below.

How ISPMan Works

The basis of ISPMan is, of course, the LDAP directory. Where standard attributes don't exist, ISPMan defines its own schema that defines the data structure the ISPMan applications use to communicate between cluster members (e.g., the location of a user's mailbox), and for communications with machines outside the cluster (e.g., DNS data).

A user record, for example, has the following basic attributes (the schema source is in parentheses):

uid (cosine.schema)
uidNumber (nis.schema)
homeDirectory (nis.schema)
userPassword (core.schema)

ISPMan currently makes use of the following schemas (the source is in parentheses):

core.schema (openldap)
cosine.schema (openldap)
nis.schema (openldap)
misc.schema (openldap)
inetorgperson.schema (openldap)
dnszone.schema (from bind9_sdb)
pureftpd.schema (from pureftpd)
ISPMan.schema (ISPMan)
RADIUS-LDAPv3.schema (from FreeRadius)

Each server in the ISPMan cluster runs the "ISPMan-agent" daemon. Its purpose is to communicate tasks to the LDAP server to execute. For example, if a node running the Web user interface manager has a request to add a user, the request gets communicated via the ISPMan-agent interface to the appropriate servers, which, in turn, provision the user on the appropriate servers. Some of the tasks it can perform are the following:

create mailbox
set mailbox quota
delete mailbox
create home directory
delete home directory

ISPMan Installation

As with many projects, the most important piece for a complicated installation like ISPMan is to plan appropriately. Some questions you should ask when planning a project like this include:

- What types of services do I want to offer?
- What equipment can I dedicate to this project?
- How much redundancy do I want?
- What level of high availability am I willing to pay for (99.9%, 99.99%, etc.)?
- How many subscribers do I want the initial system sized for? How many subscribers do I think I will have in six months? In 12 months?
- Which machines will have what function?
- What is the interface into my billing system?
- Are there any other interfaces required for add-on services, such as hosting MS Exchange services?
- Do I have resellers selling my services? What are their requirements?

ISPMan itself is relatively straightforward to install. However, due to the complex nature of these installations, the details are left as an "exercise for the reader."

Additional components such as antivirus and anti-spam functionality can be added. These can be provisioned as optional additional services in order to build incremental revenue for the provider.

How an Enterprise Might Use ISPMan

The "holy grail" that ISPMan solves for the enterprise would be the "single sign on" problem. The single sign on would essentially be the employee's LDAP directory entry in the ISPMan application. This would be used for user authentication for every application in the enterprise that required it. It could also be used for a centralized "address book" if so desired.

In addition, ISPMan could be used to provision and provide email and dialup services for an enterprise. In fact, the reseller capability would be a nifty way to enable divisions of a large organization to add their own accounts, on their own subdomain or even in conjunction with an entirely different domain! The central IT organization would enable IT or division employees to manage their own subset of accounts separate from the rest of the organization.

ISPMan could be combined with something like Bynari's InsightServer to provide calendaring, scheduling, contact-list management, etc., for a total enterprise solution at a fraction of the price of MS Exchange or Lotus Notes. The added bonus is that the solution is based on open protocols, utilizing the same "standard" interface (MS Outlook). Bynari even offers an add-on Web client that emulates much of the functionality of the MS Outlook client. While not for every enterprise, certainly the price alone makes it worth looking into.

I wish to thank Atif Ghaffar for his input into this article. Next time, please join me for another installment of ISPadmin. In the meantime, I look forward to hearing your feedback!

REFERENCES

Amavis: *http://www.amavis.org/*

Apache: *http://httpd.apache.org/*

Bynari InsightServer: *http://www.bynari.net/*

Clam Anti-Virus: *http://clamav.elektrapro.com/*

Cyrus IMAP server: *http://asg.Web.cmu.edu/cyrus/imapd/*

Ensim: *http://www.ensim.com/*

FreeRADIUS: *http://www.freeradius.org/*

Horde IMP: *http://www.horde.org/imp/*

H-sphere: *http://www.psoft.net/h_sphere2_info.html*

Introduction to ISPMan written by the author, Atif Ghaffar: *http://www.linuxfocus.org/English/September2000/article173.shtml*

ISC BIND: *http://www.isc.org/products/BIND/*

ISPMan home page: *http://www.ISPMan.org/*

Lotus Notes: *http://www.lotus.com/*

Managing ISPMan's logs: *http://nakedape.cc/wiki/index.cgi/ISPManLogStats*

MS Exchange: *http://www.microsoft.com/exchange/*

MS Outlook: *http://www.microsoft.com/office/outlook/default.asp*

OpenAntiVirus: *http://www.openantivirus.org/*

OpenLDAP: *http://www.openldap.org/*

Postfix: *http://www.postfix.org/*

ProFTPD: *http://proftpd.linux.co.uk/*

PureFTPd: *http://www.pureftpd.org/*

SAGE members list archive: *http://sageweb.sage.org/resources/mailarchive/sage-members-archive/*

Neil Schneider's write-up on installing ISPMan: *http://www.linuxgeek.net/ISPMan/*

SOAP: *http://www.w3.org/TR/SOAP/*

SpamAssassin: *http://au.spamassassin.org/*

Sphera: *http://www.sphera.com/*

vhost: *http://www.chaogic.com/vhost/*

Virtuozzo: *http://www.sw-soft.com/*

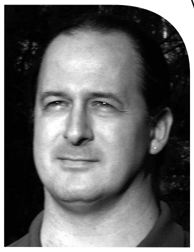
Vishwakarma: *http://www.kandalaya.com/vishwakarma.shtml*

vserver: *http://www.solucorp.qc.ca/miscprj/s_context.hc*

“eat your own dog food“

by Adam Moskowitz

Adam Moskowitz is a system administrator, programmer, manager of system administrators, certified barbecue judge, and author of the recently published SAGE Short Topics booklet *Budgeting for SysAdmins*.



adamm@menlo.com

You've no doubt heard this phrase before; you've probably even used it once or twice. As a system administrator, do you eat your own dog food? From the exact same can as your users? If not, why not?

In the late 1980s, years before the “dog food” phrase was ever uttered, I worked in Encore Computer Corporation's software development group. Not only did the operating system developers have to use the latest version of UMAX (our multiprocessor port of 4.3BSD), *everyone* in the group did, too. Talk about a good use of peer pressure! I can't prove that this practice actually improved the quality of our software, but when a user reported that something was broken, we tended to believe them, usually because we had already discovered the problem for ourselves.

When a user says, “Foo is broken,” I think the worst possible response a system administrator can give is, “It works for me.” It's even worse when the sysadmin is on a different hardware platform than most users, running a different operating system, and using a different shell. Eating your own dog food won't stop you from giving such a bad answer, but it at least mitigates your “sin.” On the other hand, I think users find it comforting when your response is, “Yes, I've noticed this, too; I'm already working on fixing it.” Eat your own dog food often enough and this is likely to become your usual response.

Here's how I think the concept should apply to system administrators:

First, use the same hardware platform and operating system as your users. If there are several, and it's not feasible to give every sysadmin in the group one of every platform, spread them evenly throughout the group. Apply the same patches to your machine as you apply to users' machines: no more and no fewer. Upgrade the OS on your machine no more than one day before you upgrade users' machines.

The obvious complaint at this point is usually, “But I have to test new patches/operating systems/whatever somewhere before we put them in production.” Yes, you do, and that's what test machines are for. Treat your desktop system as a production (user) machine and do your testing elsewhere.

Next, shells: These are very personal things, and one's setting is even more so. I've always believed that an organization should support a (limited) number of shells, but to the greatest extent possible allow users to run whatever shell they want. By “support” I mean that an effort will be made to make sure that all supported programs run on each of the supported shells, and problems in this area will be investigated (and, one hopes, fixed) by the appropriate part of the IT organization. In an ideal world, users will be able to run all supported programs with an empty shell rc file, because the system-wide default file will contain all the required settings. If that's not possible, new users should be given an rc file that looks something like this:

```
# Uncomment the next line if you plan to run ABC
# . /etc/rcfiles/abc
# Uncomment the next line if you plan to run DEF
# . /etc/rcfiles/def
```

I know it would be folly to suggest that system administrators be forced to use the same shell(s) as users. Instead, I propose that there be a test account for every shell, either without an rc file or with the same one given to new users. If it contains lines like those shown above, uncomment them as appropriate to test a given problem (or to match the users' settings), then comment them out again when you're done. Here's a concrete example of what I mean:

```
joesixc:*:99901:23:CSH test account:/home/joesixb:/bin/csh
joesixk:*:99902:23:KSH test account:/home/joesixk:/bin/ksh
joesixs:*:99903:23:SH  test account:/home/joesixs:/bin/sh
```

(I once worked on a project where "Joe Sixpack" was the name we gave to our target customer.)

If a user reports that something isn't working, and it isn't something obvious like a malformed command, log in to the appropriate account on the affected machine, modify the rc file as necessary, and only then try to duplicate the user's problem. If, after trying all this, it still "works for you," say to the user: "I can't seem to duplicate the problem, I'll have to come to your office and work with you to figure out why you're having the problem."

If you write tools, for use either by users or your fellow system administrators, you should make it a point to use those tools and not the underlying "raw" commands (if such commands exist). For example, I once wrote a set of Perl scripts to manipulate DNS files without having to use an editor, and some very simple Web forms to call those scripts using CGI. The other UNIX admins used the command line version, and the Windows and Macintosh guys used the Web forms; the idea was to allow anyone in the group to make DNS changes without having to worry (too much) about the persnickety file formats. Even though there were three of us in the group who were perfectly capable of editing the files by hand (and getting it right, at least most of the time), the agreement was that we would always use the tools I had written.

There were several interesting results from this "experiment": First, the number of times DNS broke because one of the senior admins had edited the files by hand and gotten them wrong (or forgot to increment the serial number) dropped significantly. Second, we found several bugs in the tools that the junior people wouldn't have discovered for a while (if ever). Third, we came up with several modifications to the tools that made them even more useful to the junior staff.

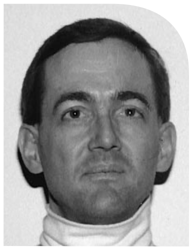
Clearly, eating my own dog food not only helped improve the quality of the tool I had written, it actually helped me reduce the number of mistakes I made while doing my job. That is, after all, the reason for writing such tools in the first place – but what good are they if you don't use them?

There are other ways in which the "dog food" concept can be applied to system administration, but I trust that you can figure them out for yourself.

If you write tools, for use either by users or your fellow system administrators, you should make it a point to use those tools and not the underlying "raw" commands.

using C# properties and static members

by Glen McCluskey



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

glenm@glenmccl.com

In this column we're going to continue our examination of the C# programming language, and look at two particular features of C# classes.

We'll start by considering the use of properties, which are kind of a hybrid between data fields and methods. We'll then go on and look at static class members.

Properties

Imagine that you're developing a C# class, and that class will have a data field that represents a calendar year. The field is set by the constructor and validated to ensure that the year is 1800 or later. You also want the ability to change the year after the fact, in existing objects of the class.

Here's some C# code that illustrates this approach:

```
using System;

public class Prop1 {
    private int year;

    public Prop1(int y) {
        SetYear(y);
    }

    public int GetYear() {
        return year;
    }

    private const int MINYEAR = 1800;

    public void SetYear(int y) {
        if (y < MINYEAR)
            throw new ArgumentException("year < " +
                        MINYEAR);
        year = y;
    }
}

public class TestProp1 {
    public static void Main() {
        Prop1 p = new Prop1(1956);
        Console.WriteLine("year #1 = " + p.GetYear());

        //p.year = 1977;
        p.SetYear(1977);
        Console.WriteLine("year #2 = " + p.GetYear());
    }
}
```

The year field is private, meaning that it cannot be set directly from outside of the class (see the commented line in Main). If the field is made public, then there's no way to validate a new value that is set. The field is instead changed via the SetYear method, and the proposed new value is checked in this method.

This approach is very common and works pretty well, but it's a little tedious to use, with every private field requiring a pair of get/set access methods.

C# offers another approach to solving this problem, using what are called properties. A property looks like a data field in an object, but access is controlled via internal get/set methods. The property can be made public and accessed like a field, but there is a layer of control that allows the class designer to interpose specific processing when the property is accessed.

Let's look at an example:

```
using System;

public class Prop2 {
    private int year;

    private const int MINYEAR = 1800;

    public int Year {
        get {
            return year;
        }
        set {
            if (value < MINYEAR)
                throw new ArgumentException("year < " +
                            MINYEAR);
            year = value;
        }
    }

    public Prop2(int y) {
        Year = y;
    }
}

public class TestProp2 {
    public static void Main() {
        Prop2 p = new Prop2(1956);
        Console.WriteLine("year #1 = " + p.Year);
```

```
            p.Year = -1977;
            Console.WriteLine("year #2 = " + p.Year);
        }
    }
```

There's still a private year field in this code, but also a public property "Year." The property can be thought of as a "virtual field." It's treated like a data field when you're programming with the class, but there are hooks such that the class can control what happens when the property's value is retrieved or set.

In the example above, when the property value is retrieved, the private year field's value is returned. When the property is set, the proposed new value, represented by the keyword value, is first checked to ensure that it's at least 1800. Then the private field is set.

Properties enable simple field access from a programmer's perspective while, at the same time, supporting data hiding so that access to private data can be controlled.

Global Variables and Static Members

C# requires that all data fields be part of a class. This restriction leads to an obvious question: How do you implement global variables, variables that can be accessed from anywhere in your application? Using such variables is not always a good idea, but we're going to assume that you really do want them for some purpose.

To answer this question, we need to consider what is meant by the concept of static class members. Normally, you define a class and then create instances or objects of the class. For example, a Point class that represents X,Y points will have various instances, such as one that represents the point 25,35. The X,Y values in the instance are called instance members.

A static member can be thought of as belonging to the class itself, and not to its instances. For example, a static data field is shared across all instances of a class. There may be one or a million instances of the class in a running application, but there will still be only one copy of the static data for the class.

Static members can be used to implement the equivalent of global variables. Here's an example:

```
// file #1

public class Globals {
    private Globals() {}

    public static int glob1 = 100;
    public static int glob2 = 200;
}

// file #2

using System;

public class TestGlobals {
    public static void Main() {
        //Globals g = new Globals();

        Globals.glob1 = 500;
        Globals.glob2 = 600;

        Console.WriteLine("glob1 = " + Globals.glob1);
        Console.WriteLine("glob2 = " + Globals.glob2);
    }
}
```

Globals is a class with two public static data members. They can be referenced by qualifying the member names with "Globals." Globals also has a private constructor, which cannot be called from outside the class. Defining a private constructor means that no instances of the class can be created. In other words, the class is used simply as a packaging vehicle for static data members.

This same approach can be used for packaging static methods, such as methods that represent self-contained mathematical functions and that have no meaning as conventional methods that operate on class instances. Let's look at an example:

```
using System;

public class CircleFuncs {
    private CircleFuncs() {}

    public static double GetCircumference(double r) {
        return 2.0 * Math.PI * r;
    }

    public static double GetArea(double r) {
        return Math.PI * r * r;
    }
}

public class TestCircleFuncs {
    public static void Main() {
        double radius = 10.0;

        Console.WriteLine("circumference = " +
            CircleFuncs.GetCircumference(radius));
        Console.WriteLine("area = " +
            CircleFuncs.GetArea(radius));
    }
}
```

CircleFuncs is a class that groups together some methods used to calculate properties of a circle, such as its circumference and area.

Note that the Main method, the entry point to a C# application, is static. It's part of a class – in the example above, the class TestCircleFuncs – but it doesn't operate on instances of TestCircleFuncs.

Constants

Constants are closely related to static members. If you're doing C# programming, how do you define groups of constants for use in your program? Let's look at a couple of examples that illustrate some of the techniques that are available:

```
using System;

enum Color {RED = 1, GREEN = 2, BLUE = 3}

public class Const {
      private Const() {}

      public const string RED = "red";
      public const string GREEN = "green";
      public const string BLUE = "blue";
}

public class TestConst {
      public static void Main() {
          Console.WriteLine("Color.GREEN = " +
              Color.GREEN);
          Console.WriteLine("Color.GREEN as int value = " +
              (int)Color.GREEN);
          Console.WriteLine("Const.GREEN = " +
              Const.GREEN);
      }
}
```

In this first example, we use an enumerated type, very similar to what C and C++ offer. This approach works as you would expect, but is suitable only for integral types.

The Const class shows how to define a group of string constants. A private constructor is once again used, so that no instances of the Const class can be created. The fields are marked as Const, which means that they're static and cannot be changed after initialization.

When you run this program, the output is:

```
Color.GREEN = GREEN
Color.GREEN as int value = 2
Const.GREEN = green
```

Here's another slightly more complicated example:

```
using System;

public class Primes {
      public const uint NUMPRIMES = 10;
      public static readonly uint[] PRIMES;

      private Primes() {}

      private static bool IsPrime(uint p) {
          if (p <= 2)
              return p == 2;
          if (p % 2 == 0)
              return false;
          uint last = (uint)Math.Sqrt(p);
          for (uint i = 3; i <= last; i += 2) {
              if (p % i == 0)
                  return false;
          }
          return true;
      }

      static Primes() {
          PRIMES = new uint[NUMPRIMES];
          uint currvalue = 1;
          for (uint i = 0; i < NUMPRIMES; i++) {
              while (!IsPrime(currvalue))
                  currvalue++;
              PRIMES[i] = currvalue++;
          }
      }
}

public class TestPrimes {
      public static void Main() {
          Console.Write("primes = ");
          for (uint i = 0; i < Primes.NUMPRIMES; i++)
              Console.Write(Primes.PRIMES[i] + " ");
          Console.WriteLine();
      }
}
```

In this example, we want to compute a table of prime numbers. We want the table to be constant and thus not mutable after it's initialized, but at the same time, we'd like to compute the values in the table at runtime, rather than actually listing them out (2, 3, 5, 7, 11, ...) in the source code.

There are a couple of techniques that we use to implement this approach. We mark the PRIMES array as static and read-only. The read-only qualifier means that the array can be modified in the constructor, but not afterwards.

We also use a static constructor, which is called when the static data members for the Primes class are initialized. The class has both an instance constructor, which is private and used to disallow creation of class instances, and a static constructor, used to initialize the PRIMES field.

The output of this program is:

```
primes = 2 3 5 7 11 13 17 19 23 29
```

In future columns we'll start looking at some broader issues with classes, such as programming with interfaces and abstract classes.

the tclsh spot

by Clif Flynt

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

clif@cflynt.com

Client Server Sockets

Previous Tclsh Spot articles described techniques for generating IP packets to simulate an attack on a firewall, while the last Tclsh Spot article described using the Expect extension to monitor a remote system's log files. This article will start to explore techniques for coordinating the attack-and-monitor activities for a firewall exerciser.

Several software architecture options exist for a system like this. The two obvious ones are a single application with attacking and monitoring subsections and a set of cooperating applications where each application provides a subset of the functionality.

A single application is conceptually simpler, since there's no need for interprocess communications. On the other hand, dealing with multiple sections that can require attention at undefined intervals is nearly as complex as interprocess communication. The real problem with a single application architecture in this case is that it limits the system to a single hardware platform. The validation application may need to run attacks and monitors from multiple sets of hardware.

Given that there will be multiple independent processes, the next question is whether they should be peers or operate in a master-slave relationship. If all the processes were identical, it would make sense to run a peer relationship. For a system where each child task has a different purpose, a peer relationship would mean that each child would need to know how to communicate with every different type of application. With a master-slave architecture, only the master needs to know how to talk to many types of applications, and the individual applications only need to know how to talk to the master. This allows the slave tasks to be simpler applications.

The last choice is whether to control the slave applications using command line arguments, pipes, or sockets. Again, the need to run on multiple sets of hardware drives the design to a socket-based client-server architecture.

Using the Tcl `socket` command to coordinate multiple tasks is fairly simple. The Tcl TCP socket implementation is possibly the easiest-to-use socket package available, and the callback mechanism used to service clients makes it easy for a server to interact with several active clients simultaneously.

Tcl uses a `channel` abstraction for I/O. A `channel` is a handle that references a source or destination for a stream of bytes. A Tcl `channel` is similar to the `FILE` pointer in C, abstracted a bit higher to include pipes and sockets.

We open either a client or server socket with Tcl's `socket` command. A client-side socket is slightly simpler, so we'll look at that first.

Syntax: `socket` *?options? host port*

`socket`	Open a client socket connection.	
?options?	Options to specify the behavior of the socket.	
	`-myaddr` *addr*	Defines the address (as a name or number) of the client side of the socket. This can be used to specify which of several Ethernet interfaces to use, and is not necessary if the client machine has only one network interface.
	`-myport` *port*	Defines the port number for the server side to open. If this is not supplied, then a port is assigned at random from the available ports.
	`-async`	Causes the `socket` command to return immediately, whether the connection has been completed or not.
host	The host to open a connection to. May be a name or a numeric IP address.	
port	The number of the port to open a connection to on the host machine.	

The `socket` command will return a channel which can be used with the `puts` and `gets` commands to send and receive data from the channel.

As a quick test of a Tcl client, we might write the code shown below, expecting to see the beginnings of a Sendmail conversation.

```
set smtpSocket [socket 127.0.0.1 25]
set input [gets $smtpSocket]
puts "READ 1: $input"
puts $smtpSocket "helo foo@bar.baz"
set input [gets $smtpSocket]
puts "READ 2: $input"
```

Unfortunately, this won't quite work. This script generates a single line of output and then hangs:

```
READ 1: 220 vlad.cflynt.com ESMTP Sendmail
8.11.6/8.11.6; Tue, 5 Aug 2003 20:48:36 -0400
```

By default, Tcl channels use buffered I/O. The example above just hangs forever with the string "helo foo@bar.baz" sitting in the client socket's output buffer, while the Sendmail server waits for input.

Tcl provides two solutions to this dilemma: the flush command, which will flush a buffer, or the fconfigure command, which allows an application to modify the behavior of a channel.

The simplest way to solve the problem is to follow each puts with a flush command. This works fine on small programs but gets cumbersome on larger projects.

Syntax: flush *channelId*

flush	Flush the output buffer of a buffered channel.
channelId	The channel to flush.

For example:

```
set smtpSocket [socket 127.0.0.1 25]
set input [gets $smtpSocket]
puts "READ 1: $input"

puts $smtpSocket "helo foo@bar.baz"
flush $smtpSocket

set input [gets $smtpSocket]
puts "READ 2: $input"
```

This generates the expected output of a simple conversation:

```
READ 1: 220 vlad.cflynt.com ESMTP Sendmail
8.11.6/8.11.6; Tue, 5 Aug 2003 20:48:36 -0400

READ 2: 501 5.0.0 Invalid domain name
```

The better way to solve the buffered I/O problem is to figure out what style of buffering best suits your application and configure the channel to use that buffering. For a challenge/response type interaction, this is probably line buffering; a character-based interactive application (like Telnet) would use no buffering, while an application moving lots of data (like an HTTP daemon) would use fully buffered I/O.

Syntax: fconfigure *channelId ?name? ?value?*

fconfigure	Configure the behavior of a channel.	
channelId	The channel to modify.	
name	The name of a configuration field which includes:	
	-blocking *boolean*	If set true (the default mode), a Tcl program will block on a gets, or read until data is available. If set false, gets, read, puts, flush, and close commands will not block.
	-buffering *newValue*	The *newValue* argument may be set to: full: The channel will use buffered I/O. line: The buffer will be flushed whenever a full line is received. none: The channel will flush whenever characters are received.

By using fconfigure to set the buffering to line mode, we don't need the flush after each puts command.

```
set smtpSocket [socket 127.0.0.1 25]
fconfigure $smtpSocket -buffering line

set input [gets $smtpSocket]
puts "READ 1: $input"

puts $smtpSocket "helo example.com"

set input [gets $smtpSocket]
puts "READ 2: $input"
```

This script generates output resembling this:

```
READ 1: 220 vlad.cflynt.com ESMTP Sendmail
8.11.6/8.11.6; Tue, 5 Aug 2003 20:51:34 -0400

READ 2: 250 vlad.cflynt.com Hello localhost [127.0.0.1],
pleased to meet you
```

A server-side socket is a little different. Rather than opening a connection to another system, a server waits until a client requests a connection to a particular port. When a client requests a connection, a new port is assigned for the conversation, and a callback script defined in the socket -server command is evaluated.

Syntax: socket -server *procedureName ?options? port*

socket	
-server	Open a socket to watch for connections from clients.

procedureName	A procedure to evaluate when a connection attempt occurs. This procedure will be called with three arguments: ■ The channel to use for communication with the client. ■ The IP address of the client. ■ The port number used by the client.
?options?	Options to specify the behavior of the socket. -myaddr *addr* Defines the address (as a name or number) to be watched for connections. This is not necessary if the client machine has only one network interface.
port	The number of the port to watch for connections.

The code to establish a server-side socket looks like this:

```
socket -server openConnection $port
```

The script that gets evaluated when a socket is opened (in this case, the openConnection procedure) does whatever setup is required. This might include client validation, opening connections to databases, configuring the socket for asynchronous read/write access, etc.

The script has three arguments appended to it before being evaluated: the handle for the new channel, the IP address of the client, and the port assigned to the client's socket.

A simple server to report the current time and close the connection looks like this:

```
#!/usr/local/bin/wish
socket -server openConnection 12345

proc openConnection {channel ip port} {
      puts $channel [clock format [clock seconds]]
      close $channel
}
```

This will open a connection, but doesn't do anything useful. A more useful server would interact with the client. The server could use the blocking gets command to wait for input, but while this paradigm works with single-client applications like Sendmail, it won't work with multiple clients, any of which might require service at any time.

Tcl supports both the linear-program flow used with a block-until-data-is-ready model, and an event-driven flow, which can be used with a multiple simultaneous session model.

The fileevent command defines a script to evaluate when data becomes available. Using fileevent guarantees that data will be available to read when the script is called, thus the application never blocks.

Syntax: fileevent *channel direction ?script?*

fileevent	Defines a script to evaluate when a channel readable or writable event occurs.
channel	The channel identifier returned by open or socket.
direction	Defines whether the script should be evaluated when data becomes available (readable) or when the channel can accept data (writable).
?script?	If provided, this is the script to evaluate when the channel event occurs. If this argument is not present, Tcl returns any previously defined script for this file event.

Setting up a file event is commonly done on the server side within the openConnection script, and on a client side, immediately after opening the socket.

```
# Server side sample openConnection with fileevent

proc openConnection {channel ip port} {
      fileevent $channel readable [list processLine $channel]
      fconfigure $channel -buffering line
}

# Client sample open socket with fileevent

set Client(sock) [socket 127.0.0.1 12345]
fileevent $Client(sock) readable "processLine $Client(sock)"
```

The last "Tclsh Spot" article described using expect to automate examining a log file. We can use the challResp procedure from that example to build a client that will automate verifying an FTP server.

The challResp procedure provides a framework for challenge/response interactions:

```
################################################
# proc challResp {pattern response info}—
#     Hold a single interchange challenge/response conversation.
# Arguments
#     pattern:    The pattern to wait for as a challenge.
#     response:   The response to this pattern.
#     info:       Identifying information about this interac-
#                 tion for use with exception reporting.
#
# Results

proc challResp {pattern response info} {
      global spawn_id
      expect {
         $pattern {exp_send "$response\n"}
```

```
        timeout  {error "Timeout at $info" "Timeout at $info"}
        eof      {error "Eof at $info" "Eof at $info"}
    }
    return "OK"
}
```

We could automate this FTP login conversation:

```
$< ftp 192.168.90.222
  Connected to 192.168.90.222.
  220 vmware2.cflynt.com FTP server (Version
          wu-2.6.2-5) ready.
  Name (192.168.90.222:clif): anonymous
  331 Guest login ok, send your complete e-mail address
          as password.
  Password:
  230 Guest login ok, access restrictions apply.
```

with this code:

```
spawn ftp 192.168.99.99
challResp "Name" anonymous "Name prompt"
challResp "word:" foo@example.com "Password prompt"
challResp "Guest login ok" " " "FTP application prompt"
```

If there is no FTP server running on 192.168.99.99, a timeout error will be generated with the string Name prompt, and if anonymous logins are not supported, the error will include the string FTP application prompt. If anonymous logins are supported, no error will be generated.

This can be expanded and generalized into a procedure that keeps a list of arguments for challResp in a list, and iterates over them until the arguments are used up, or an error is thrown:

```
############################################
# proc runTest {}—
#    Run an FTP login test
# Arguments
#    NONE
# Results
#    Returns a list of result (Success/Fail) and optional
#    failure message.
#
proc runTest {} {
    global Client spawn_id errorInfo
    set errorInfo ""

    spawn ftp $Client(IP)

    set conversations {
        "Name" "$Client(User)" "Name prompt"
        "word:" "$Client(Passwd)" "Password prompt"
        "Guest login ok" {} "FTP application prompt"
    }

    foreach {challenge response msg} $conversations {
        set fail [catch {challResp $challenge [subst
                $response] $msg} result]
        if {$fail} {break;}
    }

    array set lookup {0 "Success" 1 "Fail"}

    puts $Client(output) [list RESULT: $lookup($fail)
                $result]
}
```

By using the associative array Client to hold the IP address, username, and password, it is easy to run multiple tests with code like this:

```
array set Client {IP 192.168.99.99 User badIP Passwd
          badPasswd}
runTest
array set Client {IP 192.168.90.222 User goodUser
          Passwd goodPasswd}
runTest
```

This set of code would create a stand-alone application with a hardcoded set of tests. The script would iterate through the tests and exit.

We can convert this into a client-server application by adding a procedure to process the data that's read from the server and a few lines to open and configure the socket. The problem with this is that the script would open the socket, send an initial "Hello," and then reach the end of the script and exit.

The vwait command is the solution to this problem. The vwait command causes a script to wait until a variable changes value. The interpreter pauses at the vwait command and enters the event loop, processing events until the variable is assigned a new value. After this the interpreter continues evaluating the commands in the script.

Syntax: vwait *varName*

varName	The variable name to watch. The script following the vwait command will be evaluated after the variable's value is modified.

The simplest way to process the data from the server is to have the server always send Tcl commands, which can be evaluated in the client using Tcl's eval command.

The eval command concatenates a set of strings into a single string, and passes it to the command evaluation section of the interpreter, just as lines in a script are evaluated.

Syntax: eval *string1 ?string2...?*

*string**	Strings that will compose a command.

```
proc processLine {channel} {
    global Client
```

```
        set len [gets $channel line]

        if {[eof $channel]} {
            close $channel
            return
        }
        eval $line
}

set Client(output) [socket 127.0.0.1 23456]
fileevent $Client(output) readable "processLine
                $Client(output)"

fconfigure $Client(output) -buffering line

# Let the server know we're open for business.

puts $Client(output) ready

set Client(done) 0
vwait Client(done)
```

The server will send the client data like this:

```
array set Client {IP 192.168.99.99 User badIP Passwd
                badPasswd}
runTest
```

When the client receives data, the fileevent command causes the processLine procedure to be evaluated, which reads a line from the socket and evaluates it.

Notice the eof test after the gets. This will catch the spurious read event generated by most TCP stacks when the other end of a socket closes.

The server end of this pair includes a list of tests to run and three procedures to coordinate the tests:

openConnection
: Accepts new client connections.

processLine
: Reads data from the client. Displays results and calls runTest to start the next test in the client.

runTest
: Sends the commands to the client.

The list of tests can simply be an identifier and a set of array indices and values to be sent to the client:

```
set Server(tests) {
    {{bad address } {IP 192.168.90.223 User badIP
                Passwd badPasswd}}
    {{bad username} {IP 192.168.90.222 User badUser
                Passwd badPasswd}}
    {{good username} {IP 192.168.90.222 User goodUser
                Passwd goodPasswd}}
    {{anonymous/badPasswd} {IP 192.168.90.222 User
                anonymous Passwd badPasswd}}
    {{anonymous/goodpwd} {IP 192.168.90.222 User
                anonymous Passwd foo@bar.com}}
}
```

The openConnection procedure registers the fileevent script to evaluate whenever the client sends data, configures the channel to be line buffered, and initializes a counter to step through the tests for this client.

Note how this procedure uses an associative array index with two fields to distinguish the test counts for connections from different clients. Using multiple fields in an array index provides the same functionality in Tcl as multiple dimensioned arrays provide in C and FORTRAN.

```
proc openConnection {channel ip port} {
    global Server
    fileevent $channel readable [list processLine $channel]
    fconfigure $channel -buffering line

    # initialize the test counter
    set Server($channel.testNum) 0
}
```

The processLine procedure starts the same as the client's processLine procedure by reading a line of data and checking for an EOF condition.

The server does a rudimentary parse, looking to see if a line starts with the phrase "RESULT". If the line starts with "RESULT", it's displayed. Once data is read, the runTest procedure is invoked.

```
proc processLine {channel} {
    global Server

    set len [gets $channel line]

    if {[eof $channel]} {
        close $channel
        unset Server($channel.testNum)
        return
    }
    if {[string first RESULT $line] == 0} {
        puts "$Server(descript): $line"
    }
    runTest $channel
}
```

Finally, the runTest procedure checks to see whether there are valid tests to be run. If there are, it sends the appropriate Tcl commands to the client and updates the counter.

```
proc runTest {channel} {
    global Server

    if {$Server($channel.testNum) = [llength
            $Server(tests)]} {
```

```
            puts $channel {set Client(done) 1}
        } else {
            foreach {Server(descript) params} \
                [lindex $Server(tests) $Server($channel.testNum)] {}
            puts $channel "array set Client [list $params]"
            puts $channel "runTest"
            incr Server($channel.testNum)
        }
    }
```

This pair of procedures implements a simple test framework that can be run with different sets of data to characterize an FTP server. It's not sufficient to handle characterizing a firewall, but it's getting closer.

Sending scripts to the client to evaluate is a technique used by agent-style applications. This technique supports a great deal of customization at runtime. Tcl's eval command creates safe sandboxed interpreters, which makes it an excellent choice for exploring agent style applications.

The next "Tclsh Spot" article will look at building a server-agent architecture to perform more tests. As usual, the complete code that was described in this article is available from *http://www.noucorp.com*.

practical perl

by Adam Turoff

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

Using Object Factories

In my last column, I demonstrated how to clean up a CGI form-building program by refactoring it and using a hierarchy of modules. Each module inherited from a common application-specific Field module and implemented specialized behaviors to produce a specific kind of CGI form field. This time, I revisit the same problem and examine a different solution – object factories and factory methods.

One of the best features of Perl is the principle of TMTOWTDI: "There's More Than One Way to Do It." Even if you have only a passing familiarity with Perl, you can use it to automate a tedious task or write a small program to get your job done. You can approach Perl as a shell programmer, or as a C or Java programmer, and still get your job done.

However, Perl is a rich language unlike any other. While you are free to use idioms from shell, C, or Java to accomplish your task, using Perl idioms can help you do it faster and with less effort. My last column took a Java-flavored approach to cleaning up a CGI program. In this column, I'll look at a more Perl-flavored approach that's easier to write, maintain, and extend.

Many Ways to Do It

Consider the problem from the last column: a CGI program that needs to create an HTML form. The simple and straightforward approach is to use the CGI.pm HTML-building functions to build a Web page one piece at a time:

```
#!/usr/bin/perl -Tw

use strict;
use CGI qw(:standard);

print header('text/html');
print start_html("Test Page");

print start_form();
print popup_menu(...);
...
print submit(), reset();
print end_form();

print end_html();
```

While that approach is easy to write and easy to understand, it is also awkward and cumbersome. It is a simple translation of HTML syntax into Perl statements for building a static Web page. Modifying and debugging this program will be more difficult than necessary – programmers will need to keep a mental model of the HTML page in mind while modifying code that uses Perl syntax. Adding dynamic features to selectively display some components will turn this simple program into something complex very quickly.

The above fragment deals with two primary tasks: building the Web page, and building the form components. In my experience, the first part of this program is static and unchanging, while the second part is more dynamic. Therefore, the program can be simplified by separating the static HTML-building parts from the more dynamic form-building parts.

One way to simplify the form-building part of this program is to describe an HTML form with a list of hashes. Each hash in this list represents a single form field. Building an HTML form involves processing these field descriptions and converting them into HTML form fields as necessary. The full Web page is then created by combining the static HTML elements with these dynamically generated form fields. A program written this way might look something like this:

```
#!/usr/bin/perl -Tw
use strict;
use CGI qw(:standard);

my @fields = (
    {
        -name => "name",
        -label=> "Name",
        -size=>50,
        -maxlength=>50,
        -procedure=>\&CGI::textfield,
    },
    ## more fields here
);

...
my @rows;
foreach my $row (@fields) {
        my $sub = $row->{'-procedure'};
        push (@rows, Tr(td($row->{-label}),
                td($sub->(%$row))));
}
```

```
print start_form(),
          table(@rows),
          submit(), reset(),
          end_form();
...
```

While this approach is a step forward, it does have problems. The format for the field descriptions found in the @fields array are undocumented. They are values that will be passed to a CGI.pm function like CGI::textfield, but that knowledge is hidden dozens or hundreds of lines away in the body of the foreach loop.

This approach also leads to a lot of duplicated information. The -name and -label components contain similar values. Instead, one could easily be derived from the other, reducing an opportunity for bugs to creep in.

Ideally, these field objects should be simple to create and use. One way to do that is to create a group of Field modules to ease the process of defining fields to build an HTML form. Here is an example of what that might look like, taken from the last column:

```
#!/usr/bin/perl -Tw

use strict;
use Field;  ## pull in all the Field::* packages
use CGI qw(:standard);

my @fields = (
      new Field::Text('Name'),
      ## more fields here
);

...
my @rows;
foreach my $field (@fields) {
      push (@rows, $field->display_row());
}

print start_form(),
          table(@rows),
          submit(), reset(),
          end_form();
...
```

In this example, the interface for building a Web page is much cleaner. Creating the @fields list is done by creating a series of objects that are defined by the Field module. Each object constructor uses sensible defaults and requires a minimal amount of information. Later on, the dynamic HTML form field generation is accomplished by calling the display_row method on each field object in turn.

The Problem with Inheritance

The interface provided by the Field::* modules certainly simplifies the job of creating a dynamic Web form. It works by using a core Field class and subclasses like Field::Text to construct specific field types:

```
package Field;
use strict;

use CGI qw(:standard);

sub init_field {
      my $self = shift;
      my %params = @_;

      ## Assign the key/value pairs for this object
      while(my ($key, $value) = each %params) {
         $self->{$key} = $value;
      }
      ## Create the field name from the text label
      my $name = "\L$self->{-label}";
      $name =~ s/ /_/g;
      $self->{-name} = $name;

return $self;
}

sub display_row {
      my $self = shift;
      my $sub = $self->{-procedure};

      return Tr(td($self->{-label}), td($sub->(%$self)));
}

package Field::Text;
use base 'Field';
use CGI;

sub new {
      my $class = shift;
      my $label =  shift;
      ## Create an object
      my $self = bless {}, $class;
      ## Finish initialization
      $self->init_field(-label      => $label,
                        -size       => 50,
                        -maxlength  => 50,
                        -procedure  => \&CGI::textfield);
}
```

Field types that display multiple values share similar behaviors. The Field::Group module helps to define field types like radio groups and checkbox groups:

```
package Field::Group;
use base 'Field';
```

```
sub init_group {
      my $self          = shift;
      my $procedure  = shift;
      my $label         = shift;
      my @values       = @_;

      $self->init(-label       => $label,
                   -procedure => $procedure,
                   -values      => \@values);
}

package Field::RadioGroup;
use base 'Field::Group';
use CGI;

sub new {
      my $class = shift;

      my $self = bless {}, $class;
      $self->init_group(\&CGI::radio_group, @_);
}

package Field::CheckboxGroup;
use base 'Field::Group';
use CGI;
sub new {
      my $class = shift;

      my $self = bless {}, $class;
      $self->init_group(\&CGI::checkbox_group, @_);
}
```

Although this module hierarchy does aid in creating dynamic HTML forms, it has a Java-flavored design that leads to overly complex Perl code. In order to define three types of fields, five classes are required in a hierarchy that is three levels deep.

Extending this library isn't difficult, but it is cumbersome. Each HTML form field type requires a new class definition. Each class definition contains a package declaration, a use base declaration, and a constructor method. While none of these requirements are particularly odious, they obscure the intent: identifying the differences between textboxes, radio groups, checkbox groups, and other HTML form fields.

Using Object Factories

Looking at the code for the Field modules, there are two primary tasks that need to be solved: creating and displaying field objects. The process of creating fields is handled by a series of constructor functions, and the process of displaying fields is handled by the display_row() method in the Field class.

Each type of field object is created by a different method. Textbox objects are created by the constructor of the Field::Text class, radio group objects are created by the constructor of the Field::RadioGroup class, and so on. But there's very little difference between these objects. In fact, the only real differences between these objects are in the data they store.

Because there are no behavioral differences between these objects, there's no necessity to create multiple class definitions. In fact, all of these objects could be created through different methods in a single class. After all, there's nothing special about object constructors in Perl – they're just class methods that happen to create objects.

Refactoring the code to take advantage of this observation, we can replace the entire class hierarchy with two classes: one to display fields and one to create field objects. The new Field class is very easy to write; it contains all of the behaviors shared across field objects. Currently, this is only the display_row() method, and a basic constructor:

```
package Field;
use CGI qw(:standard);

sub new {
      return bless {}, __PACKAGE__;
}

sub display_row {
      my $self     = shift;
      my $sub      = $self->{-procedure};
      return Tr(td($self->{-label}), td($sub->(%$self)));
}
```

The rest of the magic is in an object factory class, a class that exists to create other objects. This class, FieldFactory, contains the methods for creating and customizing new Field objects, init_field() and init_group(). The init_field() method handles the bulk of the initialization and customization of a new Field object, while the init_group() method handles tasks common to initializing group fields.

Here's the start of the FieldFactory class. These two methods are almost exactly the same as the previous versions. The main difference is that the init_field() method customizes a new object, $obj, not the current object, $self (now a FieldFactory object):

```
package FieldFactory;
use CGI;

sub new {
      return bless {}, __PACKAGE__;
}

sub init_field {
      my $self = shift;
      my %params = @_;

      my $obj = new Field;

      ## Assign the key/value pairs for this object
      while(my ($key, $value) = each %params) {
            $obj->{$key} = $value;
        }

        ## Create the field name from the text label
        my $name = "\L$self->{-label}";
        $name =~ s/ /_/g;
        $obj->{-name} = $name;

        return $obj;
}

sub init_group {
        my $self        = shift;
        my $procedure   = shift;
        my $label       = shift;
        my @values      = @_;

        $self->init_field(
                -label          => $label,
                -procedure      => $procedure,
                -values         => \@values);
}
```

The remainder of the FieldFactory class is composed of factory methods which call these two init methods to create Field objects. Here is the factory method that creates textbox fields. It is almost identical to the old Field::Text constructor:

```
sub textbox {
        my $self = shift;
        my $label = shift;

        return $self->init_field(
                -label          => $label,
                -size           => 50,
                -maxlength      => 50,
                -procedure      => \&CGI::textfield);
}
```

The real benefit comes from adding new factory methods. Here are a few more which create radio groups, checkbox groups, and pop-up menus. Note that all we need here is the code. The extraneous package declarations and other magic incantations are no longer necessary:

```
sub radio_group{
        my $self = shift;
        $self->init_field_group(\&CGI::radio_group, @_);
}

sub checkbox_group{
        my $self = shift;
        $self->init_field_group(\&CGI::checkbox_group, @_);
}

sub popup_menu {
        my $self = shift;
        $self->init_field_group(\&CGI::popup_menu, @_);
}
```

With these changes to the Field module, the CGI program needs some minor changes. First, we have to create a FieldFactory object. Next, the constructor calls to create form fields need to be replaced with method calls on the factory object. The updated code looks something like this:

```
#!/usr/bin/perl -wT

use strict;
use Field;
use FieldFactory;

my $factory = new FieldFactory;

my @fields = (
        $factory->textbox('Name'),
        ...
);

my @rows;
foreach my $field (@fields) {
        push (@rows, $field->display_row());
}

print start_form(),
            table(@rows),
            submit(), reset(),
            end_form();
...
```

Conclusion

This program demonstrates there's always more than one way to do it. Simple and straightforward programs may be easy to write initially, but they can lead to readability and maintainability problems later on as they grow. Cleaning up with ad hoc data structures can help some areas of a program and hurt others.

Restructuring programs to use modules is a very big win, and there's more than one way to factor out common code into modules. One common technique is to create a hierarchy of classes to solve a problem. Another is to create an object factory instead of a class hierarchy to describe differences between objects. Each technique has its benefits and its uses. In this example, using an object factory helped simplify the implementation of an application-specific module with very little impact on the main of the program.

the bookworm

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He is Editorial Director at Matrix.net. He owns neither a dog nor a cat.

peter@netpedant.com

For about six months or so I've been reading the same book, or actually, parts of the same book, some of them several times. This is because I've been reading *The Art of UNIX Programming* as Eric Raymond's been writing it. And it has gotten better and better as a variety of folks – including (alphabetically) Ken Arnold, Steve Bellovin, Steve Johnson, Doug McIlroy, Henry Spencer, and Ken Thompson – have put in their two cents (and, in several cases, at least a dime).

If you are an experienced user of things UNIX-y, you'll really enjoy Raymond's work. If you're a newbie, you may have to really think about much of it; and if you're in the middle, you can taste, savor, and enjoy.

I'll assume that most of my readers are mid- to high-level. If so, the best place to begin this valuable book is at Appendix D, "Rootless Root." The first time I read it, I laughed so hard I got hiccups. Once you've read that, you might want to go to pp. 35–38 (in Chapter 1), because everything that I think of as important is encapsulated there. Raymond has rendered McIlroy's, Pike's, and Thompson's versions of the UNIX philosophy into a set of bullets, which might well be put on a wall near every hacker's screen.

Rather than take you through all of Raymond's valuable pages, let me just remark on how good certain pieces are, like the 20 pages of OS comparisons; the chapters on languages (and mini-languages), editors, and tools are extraordinary.

There are things I disagree with, but anyone can cavil at anything. This is a wonderful must-have. Buy one as soon as you can.

And thank you, Eric.

More OS Stuff

Last year I read and enjoyed Lucas' *Absolute BSD*, which was on FreeBSD. Lucas has now produced a (smaller) tome called *Absolutely OpenBSD*. If you're really into security and excessively paranoid, OpenBSD is the system for you. And as it's not an "easy" system, Lucas' new book is much needed. I especially enjoyed the sections on installation and configuration and on building firewalls with pf.

Gagné's *Moving to Linux* is a straightforward exposition of just how a non-hacker PC user can get rid of "the Blue Screen of Death." If you have a friend, a co-worker, a significant other, or a relative who periodically screams, sighs, bursts into tears, or asks for help, here's the simple solution. It comes with a bootable CD of Knoppix, Klaus Knopper's variant of Debian.

With *Linux Security Cookbook*, Barrett et al. have done a nice job in presenting a lot of security tools and techniques in a brief book (barely over 300 pages). I'm disappointed at the paucity of references, but the information that's actually here is first-rate.

The second edition of *Linux in a Nutshell* has lived near my desk for four-and-a-half years. The new fourth edition has just supplanted it. The fourth edition is 1.5 times the size of the second. It seems to be more than 1.5 times as useful.

Bruce Perens, the former Debian project leader, is series editor for a slew of open source books from Prentice Hall. I've read *Intrusion Detection with SNORT* and *The Linux Development Platform*, and they are of an extremely high qual-

BOOKS REVIEWED IN THIS COLUMN

THE ART OF UNIX PROGRAMMING
ERIC S. RAYMOND
Boston, MA: Addison-Wesley, 2003.
Pp. 550. ISBN 0-131-142901-9.

ABSOLUTELY OPENBSD
MICHAEL W. LUCAS
San Francisco, CA: No Starch, 2003.
Pp. 528. ISBN 1-886411-99-9.

MOVING TO LINUX
MARCEL GAGNÉ
Boston, MA: Addison-Wesley, 2003.
Pp. 348. ISBN 0-321-15998-5.

LINUX SECURITY COOKBOOK
DANIEL J. BARRETT ET AL.
Sebastopol, CA: O'Reilly, 2003.
Pp. 308. ISBN 0-596-00391-9.

LINUX IN A NUTSHELL, 4TH ED.
ELLEN SIEVER ET AL.
Sebastopol, CA: O'Reilly, 2003.
Pp. 928. ISBN 0-596-00482-6.

INTRUSION DETECTION WITH SNORT
RAFEEQ UR REHMAN
Upper Saddle River, NJ: Prentice Hall, 2003.
Pp. 256. ISBN 0-13-140733-3.

THE LINUX DEVELOPMENT PLATFORM
RAFEEQ UR REHMAN AND CHRISTOPHER PAUL
Upper Saddle River, NJ: Prentice Hall, 2003.
Pp. 320 + CD-ROM. ISBN 0-13-009115-4.

TCL/TK: A DEVELOPER'S GUIDE, 2D ED.
CLIF FLYNT
San Francisco, CA: Morgan Kaufmann, 2003.
Pp. 758. ISBN 1-55860-802-8.

PRACTICAL PROGRAMMING IN TCL/TK, 4TH ED.
BRENT WELCH ET AL.
Upper Saddle River, NJ: Prentice Hall, 2003.
Pp. 877 + CD-ROM. ISBN 0-13-038560-3.

ity. I trust the other volumes will be as good. The snort volume provides a number of useful scripts to enable you to integrate snort with Apache, PHP, etc. The *Development Platform* does an excellent job of limning just how to go about building a Linux development environment. It's accompanied by a useful CD.

Tcl Me

I've been a Tcl fan for a long time. And I admit that I am a friend both of Clif Flynt and of Brent Welch. I liked both books when they first came out. Flynt's has improved somewhat for this second edition – and those of you who read his column in *;login:* will understand where much of it has come from. Welch's book has changed tremendously since the first edition nearly a decade ago. But so has Tcl since I first came in contact with it in 1989 or 1990. Tcl is a really fine scripting language, and these books will enable you to use many enhancements, internationalization, and the toolkits.

Networking Anniversaries

by Peter H. Salus

peter@netpedant.com

As we close in on the end of 2003, I want to point out the various and sundry "Net" anniversaries this year has brought us.

In 1968, BBN received the contract to build and deploy a packet-switching network of four nodes.

In 1973 (30 years ago), the ARPANET was made up of 35 hosts; by the end of the year, Bob Metcalfe felt it necessary to warn about security problems (RFC 602, "The Stockings Were Hung by the Chimney with Care").

A decade later, the Net had grown to 575 hosts and the DCA-enforced switch to TCP/IP had been announced. A mere five years later (1988), there were 60,000 hosts. That's only 15 years ago. Right now, my best guess is that the Internet is at 300 million hosts.

In October 1983, there were 78 email/news links in the UK being fed from UKC by Peter Collinson. Also in 1983, EARN was created – a European version of BITNET. (And while I'm talking about these things, it was early in 1984 that Jun Murai initiated JUNET in Japan.)

FIDONET was also created in 1983.

IBM's VNET grew to 1001 when Reykjavik was connected in 1983.

Networking was barely a teenager, but had become a raging success all over the world.

Moving on, in 1988 the NSFNET T1 backbone became operational (1.544Mbps!). Five years later, in 1993, both PSINet and AlterNet deployed T3 backbones. And DARPA became ARPA, again.

USENIX news

USENIX MEMBER BENEFITS

As a member of the USENIX Association, you 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, Tcl, Perl, Java, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

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

ACCESS TO PAPERS from the USENIX Conferences online starting with 1993 *www.usenix.org/publications/library/proceedings/*

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, 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. See <*http://www.usenix.org/membership/specialdisc.html*> for details.

FOR MORE INFORMATION REGARDING MEMBERSHIP OR BENEFITS, PLEASE SEE *http://www.usenix.org/membership/*

OR CONTACT *office@usenix.org*

Phone: 510 528 8649

2004 USENIX Nominating Committee

by Dan Geer

Chair, Nominating Committee

geer@atstake.com

It is time for you to think about the next round of elections for the Board of Directors and for Officers of USENIX.

USENIX has eight board members: four are officers (President, Vice President, Treasurer, and Secretary) and four are board members at large. All stand for election every two years.

Organizations like USENIX have a difficult trade-off: Renewing their leadership by officially recognizing the candidates most likely to build the organization versus the free-for-all. The risk of the former is self-perpetuation of the officers more than the organization; the risk of the latter is election due to name recognition rather than skill, thus leading to irretrievable mistakes born of inexperience. In the case of the problem of self-perpetuation, the solution is term limits, and USENIX indeed has them (*http://www.usenix.org/about/bylaws.html#art4*). In the case of the problem of getting well known but inappropriate candidates, the solution is a Nominating Committee, and USENIX indeed has one of those (*http://www.usenix.org/about/bylaws.html#art7*). None of this is magic—democracy is messy, after all.

Having the right leaders for their time is hard to predict, but you have to try. More than anything else, it is short-term essential that there are enough good candidates to choose amongst: some new and some familiar, spread over all of USENIX's sub-specialties, a history of both devotion and delivery to the organization, and the room to maneuver their private life to make USENIX a priority. At the same time, it is long-term essential that the best potential officers be gotten onto the Board, so that next set of Officers can be chosen from those with prior experience at the Board level. In other words, some new Board members every two years is a good thing, along with a convincing set of Officers known for their prior accomplishment to be ready to take leadership roles.

The Nominating Committee Chair is appointed by the sitting Board. I am that Chair this time around. The Nominating Committee must be both knowledgeable and wise, and must be willing to forego office themselves. That Committee will be announced shortly. I will choose them and we will make recommendations both for Board members at large and for Officers. We will do our best and we will very much hope you follow our advice, as by the time we are done we will believe in our decisions.

If you are considering offering yourself for election by petition, please keep in mind that the Board of USENIX is real work, never trivial, and a great place to make a difference. You should have pluck, vision, the ability to work with others, and a thorough sense of some topic area that USENIX covers. You should be goal- rather than process-oriented, have a sense of humor, and be fond of the idea that you get more accomplished if you don't care who gets the credit.

Send suggestions for nominees, comments on existing Board members, or just plain questions to me: *geer@usenix.org*. Speaking as a former Board member and Officer, I can tell you that being on the Board of USENIX is absolutely, positively worth the effort.

MEMBERSHIP NEWS

SAGE Dues

You may now join SAGE for $40! While it is no longer required that you be a member of USENIX to be a member of SAGE, we hope you will remain or become a member of both organizations. Member support allows us to continue to offer some of the most highly respected conferences and publications in the industry.

Please review the benefits and dues structure listed below and on our Web site : *http://www.usenix.org/membership/*.

We will have an electronic membership renewal process in place by this Fall. Please look for renewal notices sent to you via email.

USENIX Membership Benefits

INDIVIDUALS AND STUDENTS MEMBERSHIP

- Access to the latest USENIX Conference Proceedings on the USENIX Web site: *http://www.usenix.org/publications/library/proceedings/*
- Free subscription to *;login:* - the Association's magazine, published six times a year, bringing you technical features, summaries of USENIX conferences, system administration tips, reports on various Standards activities, and book reviews. *;login:* is sent in print, and is available online a month later to all current USENIX members
- Discounts on technical sessions registration fees at all USENIX-sponsored and co-sponsored events
- Discounts on USENIX Conference Proceedings and CD-ROMs
- The right to vote on matters affecting the Association
- Savings on books, journals, and software from cooperating publishers
- 15% off subscription to *The Linux Journal*
- 20% off O'Reilly & Associates publications
- $10 off subscription to *Sys Admin*
- 20% discount off No Starch Press books

INSTITUTIONAL MEMBERSHIP (EDUCATIONAL, CORPORATE, AND SUPPORTING)

- Includes all of the benefits of an Individual membership, plus:
- All USENIX Conference Proceedings in a downloadable format to your server, so that staff and students at your institution have anytime access to all papers from our events

Supporting Members also receive:

- A free ad in *;login:* once during the membership term
- 10% discount on sponsorship and exhibit fees at USENIX events
- Ten member-priced conference registrations for your company staff during the membership term
- Your name and URL listed on our Supporting Members Web site, as well as acknowledgment in printed materials for events and in all issues of *;login:*

SAGE Membership BENEFITS

- Access to the members-only area of SAGEweb, which offers Web-based services, including employment and job placement services, a speakers bureau, a mentoring program, and a discussion site: *http://www.sageweb.sage.org*
- The ability to join SAGE-only electronic mailing lists
- The Annual System Administration Salary Survey
- Discount on the registration fee for the annual LISA Conference
- The most recent "Short Topic in Systems Administration" series booklet – practical publications covering system administration issues and techniques, with online access to all booklets: *http://sageweb.sage.org/resources/publications/short_topics.html*
- The right to vote for the SAGE Executive Committee and on other SAGE matters
- A Code of Ethics for System Administrators, suitable for framing

conference reports

This issue's reports focus on The USENIX Annual Technical Conference and on the 15th Annual FIRST Conference.

OUR THANKS TO THE SUMMARIZERS:

FOR USENIX ATC '03

William Acosta
Raya Budrevich
Francis Manoj David
Rik Farrow
Shashi Guruprasad
Hai Huang
James Nugent
Manish Prasad
Peter Salus
Benjamin A. Schmit
Wenguang Wang

FOR FAST 2003

Anne Bennett

USENIX Annual Technical Conference

June 9–14, 2003

San Antonio, Texas

AWARDS

The USENIX Lifetime Achievement Award (the Flame) was given to Rick Adams for implementing Serial Line IP (SLIP) and founding UUNET, thereby making the Internet widely accessible. In 1982 Rick ran the first international UUCP email link at the machine seismo (owned by the Center for Seismic Studies in Northern Virginia), which evolved into the first (UUCP-based) UUNET. He maintained "B" News (at one time the most popular Usenet News transport), wrote the first implementation of SLIP (Serial Line IP), and defined the first protocol for running TCP/IP over ordinary serial ports (in particular, dial-up modems). The SLIP protocol was superseded, years later, by PPP, which is still in use. Rick founded a nonprofit telecommunications company, UUNET, to reduce the cost of uucp mail and netnews, particularly for rural sites in America. (UUNET was founded with a $50,000 loan from the USENIX Association, which was subsequently repaid.) UUNET became an official gateway between UUCP mail and Internet email, as well as between North America and Europe. It hosted many related services, such as Internet FTP access for its UUCP clients and the comp.sources.unix archives. Rick spun out a for-profit company, UUNET Technologies, which was the second ISP in the United States. The for-profit company bought the assets of the nonprofit, repaying it with a share of the profits over the years. The nonprofit has spent that money for many UNIX-related charitable causes over the years, such as supporting the Internet Software Consortium. The for-profit ISP became a multi-billion-dollar company and was merged with MFS (Metro Fiber Systems, a wide-area optical-networking company), MCI, and then Worldcom, rising to challenge the largest telecommunications companies in America. He is co-author of *!%@:: A Directory of Electronic Mail Addressing & Networks*, published by O'Reilly Books. He is also co-author of RFC-850, the Standard for Interchange of USENET Messages, which was updated to become RFC 1036 in 1987.

The Software Tools User Group (STUG) award recognizes significant contributions to the community that reflect the spirit and character demonstrated by those who came together in the STUG. Recipients of the annual STUG award conspicuously exhibit a contribution to the reusable code-base available to all and/or the provision of a significant, enabling technology directly to users in a widely available form.

The 2003 award was given to CVS (the Concurrent Versioning System) and its four main authors, Dick Grune, Brian Berliner, Jeff Polk, and Jim Klingmon. Without CVS, it wouldn't be possible for any number of people to work on the same code without interfering with each other. It can be argued that without remote-collaboration tools such as CVS, most of the larger free and open source software that is available today could not have existed. While individuals can produce significant software, collaborative methods are often needed for complex and wide-ranging projects.

Dick Grune: The original author of the CVS shell script, written in July 1986, Dick is also credited with many of the CVS conflict resolution algorithms. He developed the script at the Free University of Amsterdam (Vrije Universiteit), where he teaches principles of programming languages and compiler construction. He was involved in constructing Algol 68 compilers in the 1970s and participated in the Amsterdam Compiler

Kit in the 1980s. He is co-author of three books: *Programming Language Essentials*, *Modern Compiler Design*, and *Parsing Techniques: A Practical Guide*.

Brian Berliner: Coder and designer of the first translation of the CVS scripts to the C language, in April 1989, Brian based his design on the original work done by Dick Grune.

Jeff Polk: Jeff rewrote most of the code of CVS 1.2. He made just about everything dynamic (by using malloc), added a generic hashed list manager, rewrote the modules' database parsing in a compatible but extended way, generalized directory hierarchy recursion for virtually all the commands, generalized the loginfo file to be used for pre-commit checks and commit templates, wrote a new and flexible RCS parser, fixed an uncountable number of bugs, and helped in the design of future CVS features.

Jim Kingdon: While at Cygnus, in 1993 Jim made the first remote CVS, which ran over TCP or rsh or kerberos'd rsh, and eventually over TCP/IP. The remote-CVS protocol enabled real use of CVS by the open source community; before remote CVS, everyone had to log in to a central server, copy their patches there, etc. Some years later, Jim formed Cyclic, a company which offered CVS support and development.

KEYNOTE ADDRESS

Neal Stephenson

Summarized by Peter H. Salus

Neal Stephenson – sci-fi author extraordinaire of *The Big U* (part funny, part silly, but worth it), *Snow Crash*, and *Cryptonomicon*, among others – began his keynote by remarking that Bertrand Russell and Stephen Jay Gould didn't rewrite: they had "no delete key."

He then discussed at length Antonio Damasio's *Descartes' Error* (1994), which concerns "how the brain works." In case you don't recall, Descartes separated mind and body; Damasio – and, by extension, Stephenson – think this wrong. The separation actually dates back to Plato, and Stephenson sees it as the difference between Spock and libido (Kirk or Bones).

Stephenson went on to talk of his production of *The Big U* (1984), his first opus, his creation of "steaming mountains of crap," resulting in a process of cutting and distillation. He analogized this with coding and with "distilling whiskey from beer."

When we execute there is a "foreground process" and a "background process" which goes on "all the time." That background process needs space to do what it does – "which is a mystery." What we have is "a faculty of maintaining the entire stream all the time, while accessing it only bit-by-bit at a time."

Stephenson feels that we should "kick down the Platonic model." He is forcefully against PowerPoint.

"I use a fountain pen," he remarked, and went on to say that he does not use "smileys." He thinks Larry Wall came up with something in Perl because there's "more than one way to do something."

Stephenson returned to Damasio and cited Einstein's "clear images," which are "visual and muscular," though he admitted that he wasn't certain what Einstein meant by "muscular." (I'd guess "vigorous" or "forceful" would be the appropriate gloss: the German is "kraeftig.")

Mathematical entities can be combined in an infinite number of useless forms, Stephenson said, but they are capable of leading us to mathematical truth. "Invention is choice."

Down with Plato or Descartes: "We possess an emotional marking system."

"I'm going to make no thunderous pronouncements," Stephenson pronounced. "Novelists, or coders, or philosophers, or paleontologists don't go on churning out masses of filterable stuff, but succeed by doing it right the first time.

"What we do is a much more physical kind of work and emotional kind of work than is believed. Pressure to just churn out lines of code will not lead to good things in the end."

Yep.

INVITED TALKS

Engineering Reusable Software Libraries

Kiem-Phong Vo, AT&T Labs – Research

Summarized by Wenguang Wang

Phong Vo has 20 years of experience building general purpose software libraries, which cover a wide range of computing areas such as I/O, memory allocation, container data types, and data transforming. In this talk, he showed how to build efficient, flexible, and portable software libraries using the discipline and method architecture.

Vo first pointed out that traditional standard software libraries such as malloc, stdio, curses, and libc have many problems. For example, malloc fragments memory; many container data types have multiple incompatible interfaces; inconsistent and inadequate interfaces are common; programmers often resort to hacks around problem areas, which cause problems in maintenance, porting, and performance.

Vo then discussed the characteristics of ideal standard libraries. These libraries should have good standard interfaces; address unsatisfied needs; be easy to configure, use, and upgrade; and, most importantly, have decent performance. These libraries should enable applications to tailor algorithms for specific needs and simplify library composition to optimize resource usage. Vo argued that with these ideal libraries in hand, programmers could focus on writing libraries instead of writing programs, since a program consists of libraries plus the application specifics. The productivity of programming should increase due to the maximal library reuse and minimal special code.

USENIX '03 Reception

In traditional software libraries, the handle plus operations model is often used, where the handle represents resources and the operations define resource management functions. Although this model can address immediate needs and is easy to use, it causes the problems discussed above. To address these limitations, Vo presented the discipline and method architecture (D&M), which was used when he developed the Vmalloc, Sfio, Cdt, and Vcodex libraries with other researchers.

Vo used the Vmalloc library as an example to demonstrate the D&M architecture. Vmalloc is a popular memory allocation library used to replace the standard malloc interface. In Vmalloc, a discipline defines the type of memory and the event handling of memory allocation. Typical disciplines include heap and shared memory. Programmers can define their own disciplines to extend the type of memory. A method in Vmalloc defines the memory-allocation policy. Typical methods include Vmpool, which allocates objects of the same size; Vmbest, which allocates objects based on a best-fit policy; and Vmlast, which allocates memory for a complex structure and releases all objects together in constant time.

These methods are predefined in the library and cannot be extended by programmers. They can be selected dynamically by environment variables. For example, after the VMDEBUG environment variable is set to 1, Vmalloc uses the debugging-enabled methods instead of the default-efficient methods for memory allocation, which allows various memory allocation problems to be detected automatically. This feature makes the debugging and the profiling of applications very easy whenever needed.

Vo then used the Vcodex library to show how to design a D&M library. The Vcodex library can transform data using compression/decompression, data differencing, encryption/decryption, etc. Designing a D&M library requires determining resource types and general operations, characterizing resources in a general discipline interface, and capturing algorithms/resource management in methods. In Vcodex, bytestream is identified as a resource type (i.e., discipline). All data-transforming operations (compression/decompression, encryption/decryption, etc.) are different methods since they all change some set of bytes to produce other bytes.

All the D&M libraries discussed in this talk (Sfio, Cdt, Vmalloc, Vcodex) can be downloaded from http://www.research.att.com/sw/tools. The AST OpenSource software collection built on top of some of these libraries is available at http://www.research.att.com/sw/download.

The Convergence of Ubiquity: The Future of Wireless Network Security

William A. Arbaugh, University of Maryland at College Park

Summarized by Rik Farrow

Bill Arbaugh, who often works in the wireless arena, started off his talk with a joke (I wish I would remember to do that). He showed slides of Free Software, Free Willy, Free Kevin, Free Martha, Free Wireless, and even Free Beer. But what Arbaugh really wanted to talk about was the past, present, and future of wireless security.

Hotspots are a part of the present of wireless networks. You can find maps of wireless networks, collected by war driving, for most large American cities, and some of these networks are open, requiring no authorization or encryption key. Arbaugh postulated that your next generation cell phone will work over wireless networks when possible, and fall back on slower, but more ubiquitous, cell phone technology when necessary. As you walk out of a hotspot talking on your cell phone, it will switch transparently from the Internet to the cellular network. Arbaugh quipped that the poor sound quality provided by cell phones has made short breaks in communication and occasional garbling expected in phone calls, and that will make Internet use more acceptable.

The ghost of wireless security past is WEP, or Wired Equivalent Privacy. Arbaugh described the author of WEP as being in hiding, having created a cryptographic protocol that failed in every imaginable way.

The present of wireless security is Wi-Fi Protected Access (WPA). WPA bolts on over existing hardware and solves some of the problems inherent in WEP. For example, keys will change with each packet sent, and packet integrity will actually work, due to a different set of algorithms (TKIP). Stronger access control (unlike the MAC addresses used by WEP) is included. WPA2 will follow, require hardware changes, and support AES.

Alas, WPA does not solve the current denial-of-service issues. Arbaugh demonstrated this by sending management packets (which are never encrypted or authenticated), which knocked people off the room's wireless channels.

Arbaugh's future includes smaller devices, with the transparent transitioning between hotspots and cellular mentioned earlier, IPv6 addresses and routing, and always-on connections. All devices will need personal firewalls in addition to anti-virus – not that that will protect users from the management back doors that already exist in many cell phones today.

Obviously, we are in for a lot of surprises, and interesting work, in the future of wireless.

Intellectual Property in an Age of Commerce: Core Issues in the SCO / Linux IP Suit

Chris DiBona, Damage Studios

Summarized by James Nugent

Chris DiBona gave a highly informal discussion-format talk on the SCO v. IBM suit and what it may mean. Jon Hall and Don Marti (*Linux Journal*) also participated in a semi-panel format.

First, a general discussion of the issue is in order. On March 6, 2003, SCO filed a suit alleging that IBM had stolen some code from them and placed it in the Linux kernel; they asked for $1 billion [now $3 billion – ed.] in damages. This was supposedly done as part of Project Monterey , a joint effort involving SCO, IBM, and other companies to produce an enterprise UNIX OS for the IA-64. SCO has indicated that they may go after large-scale users of Linux and mailed letters to that effect to 1500 companies. Even if IBM is exonerated, it is not clear that this would prevent SCO from bringing suits against other companies. This is a problem for small companies, which lack the money for a well organized legal defense.

Several legal issues were brought up. One of them, an obscure legal ruling from the late 1800s involving the sale of a pregnant cow when neither the seller nor the buyer knew this fact, may be used to allow SCO to continue, despite their releasing Linux code under the GPL.

SCO has not yet divulged in June where the proprietary code is. A panel of people from the open source community is forming to look at the code under a very restrictive NDA, thus making the panel of questionable value.

Jon Hall brought up the point that a comparison of the SCO and Linux code is insufficient, since code could have been inherited from a multitude of other places, such as BSD.

The impact of this suit on the future is unclear, although it could affect the willingness of companies to share code.

How to Build an Insecure System Out of Perfectly Good Cryptography

Radia Perlman, Sun Microsystems Laboratories

Summarized by Rik Farrow

Radia Perlman always seems like she is having a great time, and this talk was no exception. She selected cryptography bloopers from a much larger collection than she could present during one talk. She began with an email system based on a one-time pad. One-time pads are the strongest, most secure way of encrypting data, provided the pads themselves are kept secure and are used correctly. In the example she described, the email system used the pads with the XOR operation two times too many, making it possible to extract the pads from each message (and decrypt the messages as well).

One of Perlman's personal pet peeves is screensavers. When her screen goes blank, she immediately assumes that her screensaver has kicked in, and types her password. Of course, while this might be the case, a power-saving feature might have just blanked the screen, so if you see a weird set of characters in an email from her, it just might have been a password that she just had to change.

Perlman lambasted systems such as one where whenever you change your password, it gets emailed back to you. Or an SSL-protected online site that emails you back a receipt that includes your credit card information.

Perlman also contrasted secret and public keys. One big reason for Kerberos, quipped Perlman, was the cost of public key licenses. Kerberos requires a directory that must always be available and securely hold everyone's secrets. Public keys no longer have the onerous license fee but require Public Key Infrastructures instead.

She described the four types of PKIs: monopoly, oligarchy, anarchy, and bottom-up. In monopoly, everyone pays Verisign to play. In oligarchy, 80 or more vendors all have self-signed keys in all browsers, with no mechanism for revocation. In anarchy, we use the PGP model, which scales poorly. Bottom-up appeared the most reasonable, in which each organization has its own certification authority (CA), and organizations sign the certificates for the CAs they need to work with.

Perlman went on to rant about the craziness in the SSL certificate scheme (use of X.509), the "300 content-free pages of ISAKMP framework" for IPSec, and her top-ten favorite list of people and things that get in the way of better security. During the question and answer period, a manager begged her to lobby for better security, but Perlman stated that the world just doesn't care enough.

Alternative Top-Level Domains (aka The Game of the Name)

Steve Hotz, New.net

Summarized by Shashi Guruprasad

Steve Hotz, the CTO of New.net, gave one of the most controversial talks of this year's USENIX ATC. This was evident from the audience's hostility and bitter reactions, including a spate of profanities hurled at New.net and a few even directed at the speaker himself. New.net is a domain name registrar offering consumers a large number of new and alternative top-level domains (TLDs) – e.g., .family, .kids, .shop – that are not ratified by the Internet Corporation for Assigned Names and Numbers, or ICANN, a technical coordination body composed of a broad coalition of the Internet's business, technical, academic, and user communities. Among other things, ICANN is responsible for assignment of domain names.

New.net's position is that additional TLDs provide more choices and richer naming schemes and that a market for them exists. Currently, there is no real Internet directory and nothing to fill the gap between whois and search engines. An artificial scarcity exists and has not been mitigated by ICANN. The speaker repeatedly stressed that this is not the most important problem that needs solving in the Internet but just a place where a market exists. Some of the issues in new TLDs were: how many, Internet stability, trademarks, who will control them, and, finally, where does the money ($1 billion per year) go? New.net's perspective is that no single organization should dictate policy but that these issues should be decided by the market.

The approach that New.net has taken does not involve configuration pains for ISPs or individuals and is backward compatible with the existing system. New.net domains can be enabled either (1) via recursives that rely on US government root servers but augment with New.net domains, or (2) through user machines that have the New.net client plug-in (which still relies on the US government root servers). New.net currently offers 88 new domains. They currently have 150 million customers in total. They have tied up with five out of the top seven US-based ISPs and with many worldwide. The speaker also mentioned that their efforts to talk to ICANN and other organizations have not succeeded. They also avoid conflicting TLDs with ICANN or country-code TLDs. However, it remains to be seen if ICANN will break the Internet by launching conflicting TLDs. New.net also plans to partner with other alternative TLD providers that may bid on TLDs with ICANN, such as .kids or .golf. Some of the big companies such as Microsoft, AOL, AT&T, Verisign, and Nokia are potential new TLD providers. However, New.net is not really making an effort in this direction. The reality of "breaking the root" will occur only if multiple non-cooperating businesses promote competing namespaces. The last point was that New.net has been operational for the past two years, yet the Internet is not broken.

During the Q&A, one person expressed the wish that New.net would fail. Another mentioned that New.net has broken the fundamental principle of a single consistent naming system wherever you go. Someone else said he believed that things would break only if there was significant adoption. Another person asked how it matters whether there is a .usenix or usenix.org. The answer was that it is not a necessity but a matter of choice or desire that some people would like to have. An important counterpoint that came up was how to handle Uniform Domain-Name Dispute-Resolution Policy (UDRP) when numerous TLDs exist. For example, how would they deal with someone wanting to open up a .boycott or .sucks domain with organization names in it for people to criticize companies. It would also be harder to enforce anti-cybersquatting laws. To the speaker's position that we need a consortium rather than one organization, one person responded that ICANN was the consortium. Lastly, it was pointed out that New.net names will be invalid if DNS security extensions are implemented.

Modeling the Internet

Harry DeLano and Peter H. Salus, Matrix NetSystems

Summarized by William Acosta

As the Internet continues to grow, it becomes harder to represent the network graphically using a geographical model. Such representations of the network lack the ability to visually express such details as the location of the "big pipes," peering relations, and routing information. To illustrate the increasing complexity of the visual models of the Internet, the speakers presented a history of "maps" of the Internet, starting with the earliest known maps of ARPANET from 1969 and continuing with the geographical maps through the 1970s and 1980s that included the first satellite links to both Hawaii and Europe and the integration of multiple networks like Usenet to form what is now know as the Internet.

One of the fundamental questions that modeling attempts to answer is, how does the Internet behave? This question can be broken down into several components: What are the nodes and their connections? How can the Internet be visualized? How can its health be monitored? How can we detect and respond to events? Some of the techniques used for measurement analysis include the traceroute program, BGP information, and DNS data. Similarly, tools such as ping and the Internet Weather Report [now discontinued - ed.] are used for monitoring the "health" of the Internet. However, these tools are not perfect. As an example, the speakers noted that traceroute does not accurately reflect the existence of multiple parallel paths to some destination.

The speakers discussed several projects (from Lumeta, CAIDA, MIDS) which analyze and track certain sets of Internet information. In particular, the speakers showed the Internet averages of latency, packet loss, and reachability observed during events like the September 11, 2001, terrorist attacks and the April Fools' Virus, to demonstrate how these tools tracked the effect these events had on the Internet. Additionally, the speakers discussed tools, such as Graphviz from AT&T Labs and Walrus from CAIDA, that help visualize the topology of the Internet.

Currently, data collection is subjective and performed in an ad hoc manner. Additionally, the data sets obtained lack context and may be incomplete. As an example, it was noted that paths from traceroute and BGP data contain inconsistencies. The speakers suggested that data collection needs to be more objective and that the interpretation of the results deserves more scrutiny. To this end, the speakers suggested the formation of an international body patterned after the Centers for Disease Control that would both monitor the health of the Internet and be able to respond to events such as disasters and virus outbreaks.

During the Q&A session, John Quarterman asked what are the fixed points (analogous to cities on traditional road maps) of Internet maps? Suggestions from attendees included using Internet exchanges and the root nameservers. Someone pointed out that maps of the physical world are backed by a physical reality that is less mutable than the Internet. This prompted the question of whether we need better mapping science. Other issues raised during the Q&A included a discussion on the scalability of current measurement tools and the limitation on probing large numbers of hosts frequently.

URLs

CAIDA: *http://www.caida.org/*
Graphviz: *http://www.research.att.com/sw/tools/graphviz/*
Lumeta: *http://www.lumeta.com/*
MIDS: *http://average.matrix.net/*
Rocketfuel: *http://www.cs.washington.edu/research/networking/rocketfuel/*

Nanotechnology: As Hardware Becomes Software

J. Storrs Hall, Institute for Molecular Manufacturing

Summarized by Francis Manoj David

What is nanotechnology? It is the construction of self-replicating machines with atomic precision. Molecular manufacturing is actually a better term to describe this process. The manufacturing process is not straightforward. Because of their size it is not possible to just pick atoms and place them together. A molecule can be used as a handle to move the atom to the desired site. A bond-forming chemical reaction is used to fuse the atom to the rest of the structure.

Self-replication is a key requirement for nanomachines. This results in a sophisticated product with exponential growth in capital because of the 100% reinvestment. A parts assembly robot is used to build other robots. Just like a car factory, pipelining and convergent assembly lines are used to speed up the process.

Josh went on to illustrate applications of this technology. Nanomachines can be built to simulate molecules. One such design is "utility fog." These machines can "hold hands" with each other to form larger objects. By controlling the direction of the "hand-holding," one can create solids, liquids, or gases. Control of such machines requires nanocomputers. Nanocomputers, however, cannot be programmed using the usual techniques. The energy required to erase a bit, though small, is very significant at such small scales. Thus, these nanocomputers need to be programmed using reversible programming techniques that avoid erasing bits.

Flying cars are now possible. Airflow can be controlled over the skin of the car. Negative drag gives all the required thrust. Thus, the car can be sleek and elegant. Instant houses? At bacterial replication speeds, nanorobots can turn a pile of dirt into a house. However, there exists the problem of runaway nanorobots. There are several means to control these nanorobots. Removal of fuel can stop replication. Also, a fixed supply of replication unit kernels can control the replication.

Other interesting applications include thin skins that can insulate and protect humans. These skins can be so thin that they wrap around a single strand of hair, providing air management and sweat management! Respirocytes are yet another design for small nanomachines that function like red blood cells. They can store and release oxygen in the bloodstream as and when necessary. Space travel can also be made cheaper by the construction of an extremely tall platform using artificial diamonds. Thus nanotechnology has tremendous potential for the future of humankind.

URLs: *http://www.imm.org/Parts/*
http://discuss.foresight.org/~josh/
http://www.losthighways.org/radebaugh.html
http://www.moller.com/

TECHNICAL SESSIONS – GENERAL TRACK

ADMINISTRATION MAGIC

Summarized by Francis Manoj David

Undo for Operators: Building an Undoable Email Store

Aaron B. Brown and David A. Patterson, University of California at Berkeley

This won the General Track Best Paper award. Human error is the primary barrier to building dependable systems. A small mistake such as misconfiguring an email virus scanner can result in lots of lost email. This damage could be prevented if operators were able to undo their mistakes. This paper presents an implementation of undo capabilities for an email distribution system.

Three Rs – rewind, repair, and replay – are necessary to achieve such capabilities. Rewind brings back a system to a time in the past, repair fixes the mistake, and replay ensures that new information is not lost. Paradoxes can arise during a replay, though. This is because the system might make changes to message bodies or restore missing email to a mailbox. To explain this to the user, explanatory emails are sent.

The architecture for the undoable email store consists of rewindable storage and a proxy that intercepts user events. All user events are encoded as "verbs" which are logged. An undo manager is used to control all undo operations. Studies with a Java implementation show that the time and space overheads are not significant. Responding to a question at the end of the presentation, Aaron also clarified that the explanatory messages themselves are not treated as user events and hence no verbs are generated for them.

Role Classification of Hosts Within Enterprise Networks Based on Connection Patterns

Godfrey Tan, John Guttag, and Frans Kaashoek, MIT; Massimiliano Poletto, Mazu Networks

This paper presents methods to automatically group hosts in a network based on their communication patterns. Grouping of hosts can simplify network management and can also detect anomalous conditions. For example, a developer's machine behaving like a manager's machine would be interesting information to an administrator.

The proposed scheme uses probe hardware to sniff all network traffic. A central aggregator then collects the information from all the probes. It is responsible for maintaining a "neighbor relationship" between hosts that communicate regularly. Groups are then identified based on host similarities. Similarity is defined as the number of common neighbors. Group formation is treated as a graph theory problem. Each node in the graph is a host and each edge with weight *e* represents the fact that there are *e* common neighbors between the hosts. For a given value of similarity, say *k*, a *k*-neighborhood graph is constructed with edges of weight *k*. Bi-connected components in this *k*-neighborhood graph are considered separate groups. Special cases are used when the graph is a tree. Groups can also be merged into a larger group based on group similarities. For more details on this, please refer to the paper.

The effectiveness of this scheme has been studied by comparing the groups generated automatically with groups of hosts determined by system administrators. Results of this study show that the groups extracted automatically closely match the desired groups. This scheme shows that automatic role classification of hosts based on their communication patterns is feasible. Mazu Network's PowerSecure software now incorporates refined versions of most of the algorithms described in this paper.

A Cooperative Internet Backup Scheme

Mark Lillibridge, Hewlett-Packard Labs; Sameh Elnikety, Rice University; Andrew Birrell, Mike Burrows, and Michael Isard, Microsoft Research

Backup service providers on the Internet are costly. This paper describes a scheme by which individual computers on the Internet can cooperate to back up each other's data. Each computer uses a set of partners to store its backup data. In return, it holds part of each partner's backup data. By making redundant copies of the backup on multiple computers, a high level of reliability is obtained. Thus, inexpensive backups can be obtained.

The backup scheme depends upon cooperation between hosts. There are always bound to be hosts that refuse to cooperate. In its basic form, this scheme has several pitfalls. Hosts may promise to hold data and not keep their word. This is discouraged by several mechanisms, including periodic challenges to ensure that partners are cooperating and a novel method called "disk-space wasting" designed to make cheating unprofitable. Attacks aimed at disrupting the service are also possible in the basic scheme. The paper outlines some solutions to prevent these attacks as well.

Results from an initial prototype show that these techniques are feasible as far as performance is concerned. The costs involved are also quite low compared to other Internet backup options.

POWER

Summarized by Hai Huang

Currentcy: A Unifying Abstraction for Expressing Energy Management Policies

Heng Zeng, Carla S. Ellis, Alvin R. Lebeck, and Amin Vahdat, Duke University

A system-level energy management technique is discussed. Energy is classified as yet another resource that can be managed by the operating system. An energy quota is allocated to each process periodically, and when each process accesses an energy-consuming hardware device (e.g., disk, CPU, network interface), it is charged the amount of energy that was spent accessing the device. When all its energy quota is spent, it cannot execute further until the operating system replenishes its quota. The authors were able to show that the system is able to achieve desired runtime, if it is possible, with a high success rate. By managing the energy quota of each application, it is easy to suppress the execution of the less important tasks, while giving more energy to the more important ones to keep the system running longer when the energy supply is low.

Design and Implementation of Power-Aware Virtual Memory

Hai Huang, Padmanabhan Pillai, and Kang G. Shin, University of Michigan

A novel technique to reduce the power dissipation in the DRAM was presented. As workloads become more data-centric, more energy is spent to sustain the ever-growing DRAM in systems. The intuitive basis of current power-management technology is that processes execute one at a time, and when each is executing, it only uses a small fraction of the total system memory. By figuring out the memory nodes each process uses, energy can be saved by putting unused memory nodes into a low energy state. Proactively allocating physical pages to minimize the number of memory nodes each process uses creates opportunities to power off a large percentage of nodes in the system, thereby saving a significant amount of energy. The authors introduced other techniques – library aggregation and page migration – to achieve energy savings. These techniques are compatible with various DRAM architectures, including SDR, DDR, and RDRAM.

GET VIRTUAL

Summarized by Hai Huang

Operating System Support for Virtual Machines

Samuel T. King, George W. Dunlap, and Peter M. Chen, University of Michigan

The authors describe several techniques to reduce the virtualization overhead associated with type II virtual-machine monitors (VMM) so they can achieve performance similar to type I VMMs. In their experiments, they used UMLinux as the guest operating system. One of the techniques they use to improve performance is to move the VMM from the user-space to the kernel, so they can avoid high-overhead ptrace calls and reduce the number of context switches when guest system calls/signals are invoked. Other techniques include the use of segmentation registers to reduce overhead of guest user-to-system and guest system-to-user boundary crossing, and the use of multiple address spaces for the virtual machine to reduce the guest user-to-user switching overhead. The performance of their system was comparable to that achieved by VMware Workstation.

A Multi-User Virtual Machine

Grzegorz Czajkowski and Laurent Daynès, Sun Microsystems; Ben Titzer, Purdue University

This paper describes the implementation of adding multi-user capability in a multi-tasking virtual machine (MVM). A MVM allows a user to execute multiple processes in the same Java Virtual Machine (JVM), and by sharing resources among these processes, it reduces resource consumption and improves overall performance. Multi-user MVM takes this a step further and allows multiple users to safely execute different processes on the same MVM. As if executing in a traditional OS, processes, files, and other resources belonging to the same user will not be illegally accessed by another user. MVM-2 extends MVM by having a separate instance of Jlogin per user session to manage user identity, environment, and other access control information. It was shown that for a typical workload, MVM-2 only incurs a small performance overhead over MVM, while providing a safe multi-user environment.

NEEDLES AND HAYSTACKS

Summarized by Wenguang Wang

A Logic File System

Yoann Padioleau and Olivier Ridoux, IRISA / University of Rennes

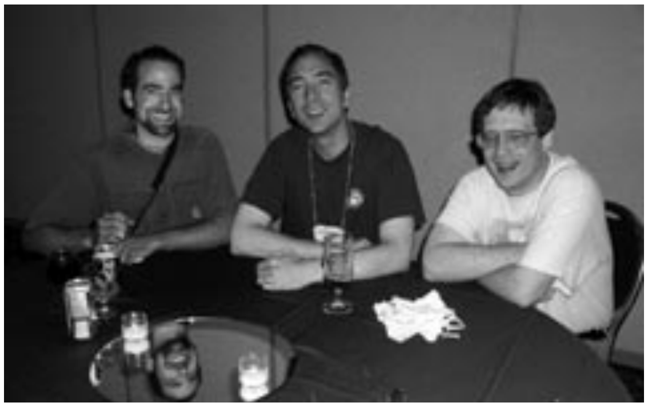
Organizing information using the hierarchical paradigm, as is done by traditional file systems, can provide navigation functions but is rigid in that an object can only be reached via one path. The Boolean query paradigm used by search engines provides flexible keyword-based search capability but lacks a navigation mechanism. In this talk, Yoann Padioleau presented a logic file system (LISFS) based on the Boolean query paradigm but extended to provide navigation. Each file in LISFS is associated with a conjunction of properties of interest. The directories are used to represent properties. One can navigate in LISFS by giving the desired or undesired properties as directory names.

A prototype of LISFS has been implemented by the authors. The data structure of the current implementation is optimized for searching and navigation. It is somewhat slow for file and property creation. The performance experiments of the prototype showed efficient navigation execution but 4 to 34 times longer creation time than ext2. The disk space overhead is 2–5KB per file, given 50 properties. A prototype of LISFS and more information on this project can be downloaded at *http://www.irisa.fr/LIS*.

APPLICATION-SPECIFIC DELTA-ENCODING VIA RESEMBLANCE DETECTION

Fred Douglis and Arun Iyengar, IBM T.J. Watson Research Center

Delta-encoding is a data-compressing method that represents an object using its difference relative to a similar object. In this talk, Fred Douglis showed how to use delta-encoding to compress a set of files without specific knowledge of these files in advance. Unlike previous approaches, the resemblance between files is detected dynamically.


Ted Hayelka, Jim Larson, and Bart Massey enjoying the USENIX Reception

Douglis first explained how to detect resemblance between objects. The Rabin fingerprints are used to compute hashes of overlapping sequences of bytes in a file. A subset of these fingerprints (i.e., features) are used to represent the file. Files sharing many features would, hopefully, have similar contents.

After two objects are detected as similar, delta-encoding is applied to compress the objects. The authors evaluated a number of parameters of this approach. They found that delta-encoding using Rabin fingerprints can improve on simple compression by up to a factor of two, depending on workload. A small fraction of objects can potentially account for a large portion of the space and bandwidth savings. More importantly, when multiple files match the same number of features, any file can be used as a good base for computing delta.

OPPORTUNISTIC USE OF CONTENT ADDRESSABLE STORAGE FOR DISTRIBUTED FILE SYSTEMS

Niraj Tolia, Mahadev Satyanarayanan, Carnegie Mellon University and Intel Research Pittsburgh; Michael Kozuch, Brad Karp, Intel Research Pittsburgh; Thomas Bressoud, Denison University and Intel Research Pittsburgh; Adrian Perrig, Carnegie Mellon University

A distributed file system on WAN is often slow. Niraj Tolia presented their work on building a distributed file system, called CASPER, which exploits the readily available Content Addressable Storage (CAS) to cache the objects in a remote server so that the read traffic through WAN can be reduced.

File recipes, which are the file content hashes that describe the data blocks composing the file, are computed for each file and stored in the server. When a client wants to read a large file, it first fetches the recipes of this file from the server. The client then uses these recipes as keys to search the file blocks in CAS. If a block is found, the client retrieves it from CAS; otherwise, the client reads it from the server. The client finally reconstitutes the full file content using these blocks. The experimental results of CASPER showed that when client-server bandwidth is low, CASPER achieves a significant reduction of runtime when reading large files from the server, given that a significant portion of the file blocks can be found in the CAS.

CHANGE IS CONSTANT

Summarized by Shashi Guruprasad

SYSTEM SUPPORT FOR ONLINE RECONFIGURATION

Craig A.N. Soules, Gregory R. Ganger, Carnegie Mellon University; Jonathan Appavoo, Michael Stumm, and Kevin Hui, University of Toronto; Robert W. Wisniewski, Dilma Da Silva, Orran Krieger, Marc Auslander, Michal Ostrowski, Bryan Rosenburg, and Jimi Xenidis, IBM T.J. Watson Research Center

Craig Soules spoke about their work to extend and replace active OS components to perform an online reconfiguration. OSes are complex pieces of software that are required to work under a variety of hardware resources and usage patterns. To overcome this problem of one size fits all, OSes are required to be tuned with the right set of components to work well under many demanding conditions. The need to bring down the system in order to reconfigure it is costly in terms of availability, human time, and lost system state. The goal is to perform such online reconfiguration with minimal overhead. Implementing this in a traditional OS is difficult because of unstructured component boundaries, and thus it is hard to backfit new code. An object-oriented OS such as IBM's K42 is a suitable fit and is used for this purpose.

To support online reconfiguration, component boundaries must be clearly defined, external references to the component must be updated, state transfer mechanisms must be defined between the components being replaced, and components should be quiesced during reconfiguration so as to avoid state corruption. An object translation (indirection) table approach is used to take care of external references. Components handle their own state transfers, since the states of different components vary. An interposition phase interposes a mediator around an existing object that is being reconfigured. The mediator performs the hot-swapping of the object in three phases: forward, block, and transfer. It uses a thread-generation mechanism to detect active calls, and eventually blocks new calls and thus detects when a quiescent state is reached. Once new calls are blocked, the new object is hot-swapped and a state transfer is initiated. After this process, the mediator is detached, the old component is destroyed, and all new calls directly occur on the new component.

The performance overhead was around 500 CPU cycles for the interposition phase and 4000 cycles for the hot-swapping phase on an IBM RS/6000 24 600Mhz Power-PC CPU-based server. Two benchmarks, Postmark and SDET, were used for the evaluation. Several well-known adaptive algorithms performed the online reconfiguration to demonstrate the utility. Some of the open issues include coordinated swapping and recovery from broken components that replace working ones.

Checkpoints of GUI-Based Applications

Victor C. Zandy and Barton P. Miller, University of Wisconsin

Victor Zandy talked about transparently migrating the GUI (X Windows) belonging to unmodified applications between hosts without premeditation, a system that he calls "guievict." GUI replication as well as process migration is also possible using this system. Some of the related work such as VNC or xmove needs premeditation and uses proxies to achieve GUI migration.

The interesting part of the system is the use of program-editing techniques to introduce new code into a running application to achieve the goal. A tool with an architecture-independent API known as Dyninst is used for "hijacking" an application. It's easier to hijack a dynamically linked library than a statically linked one, which requires introducing the dynamic linker code into the application's process space. In order to achieve a high level of transparency, the system tries to automatically determine the original X-server host so as to redirect communication to the new X-server. This is achieved by enumerating the process's socket descriptors that are communicating with a peer port that falls in the well-known X-server port range. This could fail, however, if tunneling is used, and is remedied by trying to determine the right socket by a brute-force approach which involves connecting with every port that the application is already connected to. A user could also explicitly provide this information. Another step in the process is to find a suitable point in the X message stream so as not to corrupt or miss any message. This is performed by walking through the process stack in search of X library stub functions and, if any exist, repeating the operation after a timeout. A limitation of this approach currently is that it does not work with stripped statically linked binaries, for which alternate mechanisms are being worked out. The last step is to retrieve the list of GUI resources that exist on the source X-server and re-create them on the target X-server. X-fonts present a problem because of lack of enough state on the application side to determine the font names. The current solution is to retrieve all fonts and match the font geometry with the ones being used. This hideously slow approach will likely be solved with client-side fonts in the future. Currently, this overhead is mitigated in subsequent requests by caching font data.

URL:
http://www.cs.wisc.edu/~zandy/guievict/

CUP: Controlled Update Propagation in Peer-to-Peer Networks

Mema Roussopoulos and Mary Baker, Stanford University

Mema Roussopoulos presented work addressing the problem of reducing search query latency in peer-to-peer (P2P) systems. Search queries are a performance bottleneck in both structured (Chord, CAN, Pastry, Tapestry) and unstructured (Gnutella, Freenet) P2P systems. Recent work has used caching of metadata that is returned in response to these queries at intermediate nodes along the path and is referred to as Path Caching with Expiration (PCX). PCX mechanisms are only a partial solution, as there is no maintenance of the caches along the path.

CUP addresses this problem by having asynchronous propagation of metadata updates with independent local node policies rather than global ones. It is independent of the underlying search algorithm and uses incentive-based policies. A node propagates an update only if it has an incentive to do so. For example, a node would be willing to receive updates as long as there are queries for a particular piece of metadata, since the node would have to propagate the query further if it did not maintain the cache locally. Two policies are defined, probabilistic and history-based. Separate logical query and update channels are maintained and queries are coalesced. The number of overlay round-trip hops for queries to come back with answers starting from the originator is used as a metric for the cost of a query. Miss cost is defined as the total number of hops incurred by all misses. As long as the difference in miss costs between PCX and CUP is larger than the overhead of update propagation, CUP recovers its cost.

The Stanford Narses P2P flow-simulator is used to evaluate the algorithm with a variety of cut-off policies, network scales and topologies, outgoing update capacity, and query arrival distributions. Based on their model, CUP recovers its cost by a factor of 2 to 300 under a variety of workloads described in detail in the paper. The various cut-off policies have comparable performance when the query rates are high, while second-chance history-based policy is better with low query rates.

SECURITY MECHANISMS

Summarized by Rik Farrow

THE DESIGN OF THE OPENBSD CRYPTOGRAPHIC FRAMEWORK

Angelos D. Keromytis, Columbia University; Jason L. Wright and Theo de Raadt, OpenBSD Project

Angelos Keromytis described the rationale and mechanisms behind the API created within OpenBSD, and adopted by FreeBSD and NetBSD, for integrating hardware cryptographic devices. The basic concept was to provide an abstraction layer so that programs and kernel routines could ignore the differences between devices, or even the lack of a device. Previous implementations could stall the CPU waiting for a device to respond.

Within the kernel, three functions handle creating, dispatching, and freeing a `crypto_session`. The dispatch call includes a callback parameter, so that the crypto device can progress asynchronously. Out in user-land, `/dev/crypto` provides a uniform entry point, controlled via `sysctl()` calls. The current version of OpenBSD Cryptographic Framework (OCF) is synchronous at the user level.

Another big goal of OCF beyond transparency is performance. The new API does add overhead, but in systems with hardware cryptographic devices it improves performance because the hardware devices can operate in parallel with the CPU. Future work includes smarter load balancing, algorithm chaining within the kernel thread, using the second processor as a cryptographic device in dual-processor systems, and minimizing copying overhead.

NCRYPTFS: A SECURE AND CONVENIENT CRYPTOGRAPHIC FILE SYSTEM

Charles P. Wright, Michael C. Martino, and Erez Zadok, Stony Brook University

Charles Wright presented this paper by explaining motivation (providing protection to data) and describing prior work. He mentioned that Matt Blaze's CFS suffered from network and data copy overhead, and I watched Blaze, just a few rows ahead of me, sit up a bit straighter at that statement. Wright also mentioned TCFS, Microsoft's EFS (which does not work over a network), Cryptfs, a proof of concept by the same authors, BestCrypt (commercial), and StegFS.

Design goals include strong encryption; convenience for users, system administrators, and programmers; and performance. NCryptfs has three sets of players: system administrators, who set up NCryptfs through mounts but do not hold keys; owners, who hold encryption keys; and others, called readers and writers, to whom access is delegated by owners. All keys are stored using a long-term key, and each owner uses a passphrase (currently) to authenticate and access his or her own keys.

Borrowing from Blaze's CFS, NCryptfs also uses an attach to set up encrypted directories for owners. An attach is like a lightweight user-mode mount that, unlike a regular mount, cannot hide files or directories under a mount point. Only data gets encrypted, not the metadata, so any underlying store can be used (local disk, NFS, or CIFS). In an interesting addition to this scheme, owners may delegate their access to individuals and to arbitrary groups, simply by adding authorizations for several individuals. And this delegation can even override permissions found in the underlying file system (when mounted with VFS bypass permission).

NCryptfs outperformed CFS, TCFS, but not BestCrypt in the Am-utils benchmark. In an I/O-intensive benchmark, NCryptfs was fastest. Future work includes better key management, lockbox mode, centralized key servers, and threshold secret serving.

Matt Blaze was on his feet even before Wright finished. Blaze asked if Wright was sure he didn't have the CFS and TCFS performance reversed? Wright said that they fixed some problems with TCFS that had a performance impact. All three crypto file systems were using Blowfish, but had different overhead outside of encryption.

Marc Stavely asked, Why not use underlying file system checks? Wright said that there are no ACLs on lower-level file systems, and that their ACLs provide the key needed to read/write files.

A BINARY REWRITING DEFENSE AGAINST STACK-BASED BUFFER OVERFLOW ATTACKS

Manish Prasad and Tzi-cker Chiueh, Stony Brook University

Manish Prasad explained that the goal was to instrument a program to defend against stack-based buffer overflow attacks without having access to the source. There are tools available, such as Etch, that can perform binary code translations, but the authors are not aware of any that perform return address defense (RAD) based on binary rewriting.

The authors use both linear instruction disassembly and control flow analysis. Because Intel processors use variable-length instructions, and 248 out of 256 bytes are legitimate starting points for instructions, linear analysis has real problems distinguishing code from data. The authors then follow with a second pass, starting with the code's entry point (as found in the program header), and following all function calls and branches. Even here there are problems, as GUI-based applications tend to use callbacks instead of direct function calls.

RAD replaces the prologue and epilogue of functions with its own instructions, storing the return address during the call and checking it when the function returns. They instrumented several Microsoft applications, and estimate that they missed less than 1% of functions. When used against open source Windows applications, such as gzip and Wget, they obtained similar results, but not with Apache, which includes functions without any absolute addresses (called from tables).

Overhead for an instrumented binary varied 1–4%, and a test of an exploit against an RAD-protected version of winhlp32.exe worked. Prahad ended his last slide with a note: I'm looking for work! One person asked if safe languages would help. Prahad answered that that is the best solution. A person from CERT asked about how RAD handles detected changes. Prahad said that the program exits.

FAST SERVERS

Summarized by Manish Prasad

Kernel Support for Faster Web Proxies

Marcel-Catalin Rosu and Daniela Rosu, IBM T.J. Watson Research Center

The paper addresses the challenges in Web proxy design that arise out of a large number of TCP connections. Handling a large number of network connections causes the proxy server to incur huge overheads due to context switches and user/kernel data copies. The paper presents two mechanisms to ameliorate this overhead, namely, user-level connection tracking and data-stream splicing.

User-level connection tracking allows an application to coordinate its non-blocking I/O operations with significantly fewer system calls. It is implemented by exposing certain elements of a connection's state to user-space over a piece of memory shared between kernel and user-land. The amount of shared memory required is just a small fraction of a typical Web server main memory (1MB, for application with 65K concurrent connections, 16 bytes per connection). The authors implemented a user-level select() wrapper called uselect(), which reads from the shared memory area whenever possible, and calls select() only when the required information is not available in user-space.

The other mechanism proposed is data-stream splicing, which allows forwarding of data between server and client streams from within the kernel, thus reducing a lot of data-copy and context-switching overhead. The two connections (to client and to server) are spliced at the socket level. The novel features in this mechanism are support for request pipelining and persistent connections, content caching decoupled from client aborts, and efficient splicing even for short transfers.

The proposed mechanisms were evaluated on a testbed comprising commodity hardware. Experiments were done using the Squid proxy server and benchmarked using PolyGraph. The speaker presented CPU utilization results for different request rates for a benchmark biased toward small files, which showed best overhead reduction with a combination of uselect and splicing. Splicing achieved very good results for large objects. Also, the proxy hit response times showed substantial reductions using a combination of uselect and splice (10–30%). However, the miss response times reduced only by about 0.5 to 1.3% using the same combination.

Multiprocessor Support for Event-Driven Programs

Nickolai Zeldovich, Stanford University; Alexander Yip, Frank Dabek, Frans Kaashoek, and Robert T. Morris, MIT; David Mazières, New York University

The contribution of this paper is libasync-smp, a library that allows event-driven applications to take advantage of multiprocessors by running code for event handlers in parallel. Typically, event-driven programs are structured as a collection of callback functions which a main loop calls as I/O events occur. However, to execute callbacks concurrently on a multiprocessor requires running multiple copies of the application or fine-grained synchronization.

This paper maps this problem to graph-coloring by allowing the programmer to choose a color for each callback, so that callbacks with the same color are never executed concurrently. Thus, event-driven servers can get more juice out of a multiprocessor machine with minor modifications in code. As proof of concept, they modified the SFS file server (about 90 lines of changed code out of a total of about 12,000), and observed that the modified version on a four-CPU server ran 2.5 times as fast as an unmodified SFS server running on one CPU. Further improvements were observed in task-processing rates with thread-affinity optimizations.

BIG DATA

Summarized by Manish Prasad

Seneca: Remote Mirroring Done Write

Minwen Ji, Alistair Veitch, and John Wilkes, Hewlett-Packard Labs

Minwen Ji presented Seneca, a prototype remote-mirroring storage system. A primary contribution of the paper is a comprehensive taxonomy of design choices for remote mirroring over various axes: fault-coverage (component-failure tolerance), degree of synchronization (divergence), update propagation and acknowledgment, and location of data duplication. They also classify related work in the area, based on their taxonomy.

The second contribution of the paper is the design of a robust asynchronous remote-mirroring protocol that supports write coalescing, asynchronous propagation, and in-order delivery and that provides resilience to many kinds and sequences of failures, low network bandwidth demands, and low (and tunable) data loss. The principal goal of Seneca's design is to make the data available and keep each copy consistent despite disk array failures, Seneca box failures, network failures, and both temporary and permanent site outages. Its levels of fallback divergence modes are time-bounded, log-space-bounded, and unbounded. Updates are propagated either atomically in order, asynchronous batched, or out of order. In Seneca, the data duplication is done in the duplexed SAN appliance.

Seneca's correctness is verified using a simulator which "approximates" the I/O automaton model. One of the key goals of performance evaluation experiments is to answer the question as to how much network bandwidth can be saved by delaying I/Os (asynchrony) and coalescing writes. Results show that substantial reductions in WAN traffic (5–40%) were observed for a batch size of 30 seconds.

EVICTION-BASED CACHE PLACEMENT FOR STORAGE CACHES

Zhifeng Chen and Yuanyuan Zhou, University of Illinois at Urbana-Champaign; Kai Li, Princeton University

The work addresses the problem of cache placement (not replacement) in the context of buffer cache management for lower-level caches (resident on the back-end storage server) in a multi-level storage hierarchy (client-file server-disk array). Cache placement typically follows an access-based policy, which places a block into a cache when this block is accessed, so that the block that resides in the upper-level cache is also contained in the lower-level cache. While this might be essential when upper-level caches are significantly smaller than the lower-level caches, it's not quite so important in a storage cache hierarchy where storage-server and file-server (storage client) caches are comparable in size. An eviction-based strategy places a block in the cache only when it is evicted from an upper level.

The paper uses idle distance as a metric for evaluating the effectiveness of a cache-placement policy. Idle distance is the period of time the block resides in the cache without being accessed. A better cache-placement policy will have lower average idle distances. The paper presents real-world storage-access traces, which reveal that eviction-based strategies show lower average idle distances than access-based policies. The speaker presented simulation results comparing the two placement policies as applied in combination with various cache-replacement strategies (LRU, frequency based, 2Q, MQ), which showed that the eviction-based strategy always performed better.

The work proposes a mechanism to transparently obtain eviction information from client buffer caches, which is nontrivial because a client buffer cache always silently evicts a clean page and only writes out dirty pages to the back-end storage system. The main idea is to maintain a mapping between buffer addresses and the disk block that it houses, so that by intercepting read/write I/O operations, any changes detected in the mapping can be attributed to the block being evicted from the cache. Evaluation on real systems showed a 22% improvement in cache hit ratios and a 20% improvement in the transaction rate.

FAST, SCALABLE DISK IMAGING WITH FRISBEE

Mike Hibler, Leigh Stoller, Jay Lepreau, Robert Ricci, and Chad Barb, University of Utah

Robert Ricci talked about Frisbee, a prototype disk-imaging system from Utah. The paper motivates the use of raw disk imaging over differential update which operates above the file system. Advantages offered by disk imaging include generality across file systems, robustness to file-system corruption, and speed. Disk imaging, however, is low on bandwidth efficiency. Frisbee (derived from "flying disks") incorporates file-system-aware data compression, data segmentation, and a custom application-level reliable multicast protocol.

The talk proceeded with a description of the Emulab environment, where the prototype was built and tested, which comprises a cluster of 168 commodity PCs. Being a time-shared setup (in the true sense of the term), every user has root access to all machines in the cluster. This necessitates returning the cluster nodes to a known state with change of users. Thus a system like Frisbee turns out to be a compelling requirement in an environment like Emulab.

Frisbee was evaluated for scalability with an increasing number of cluster nodes (from 1 to 80). Experiments were also conducted to evaluate scalability in the presence of various percentages of packet loss. Frisbee has been in production use for over 18 months. Speed was reported as the single most attractive feature of Frisbee, being able to write 50GB of data to 80 disks in 34 seconds. On this note Ricci quoted colleague Mike Hibler: "Earlier I could go for lunch while the disk reloads, but now I can't even make it to the bathroom!"

FREENIX TRACK

NETWORK SERVICES

Implementation of a Modern Web Search Engine Cluster

Maxim Lifantsev and Tzi-cker Chiueh, Stony Brook University

Summarized by William Acosta

The authors presented the design and analysis of Yuntis, a prototype for a scalable and extensible Web search engine. The design of Yuntis attempts to provide performance by using an event-driven model with one main thread of execution and "select"-like functionality to avoid the context-switching overhead of a multi-threaded design. Similarly, Yuntis employs custom disk data storage and intra-cluster communication mechanisms.

The overall process of creating the search database involves crawling the network to retrieve documents, which are compressed and stored upon successful retrieval. A page quality score is iteratively computed, keywords are extracted, and an index is created of the extracted phrases. The prototype implementation consists of 12 cluster nodes and a data set of 4 million crawled pages, with statistics and information stored in 121 separate data tables. The content is partitioned over the entire cluster. Future work includes workload balancing of the cluster, scalability improvements, and fault-tolerance mechanisms.

URL: *http://www.ecsl.cs.sunysb.edu/yuntis/*

MAIL

Summarized by Raya Budrevich

ASK: Active Spam Killer

Marco Paganini

Currently, there are three main prevention approaches to the problem of unwanted commercial email: real-time black hole lists, keyword identification, and distributed anti-spam networks. Each of these methods has many drawbacks, however, including high percentages of false negatives. ASK proposes a new solution to the problem by using a challenge-authentication model. It applies authentication to the email without interpreting the content and focuses on validating the senders rather than the message; this approach is highly effective, since spammers use fake addresses to hide their identity. The system consists of three lists: white, ignore, and black. When a message is received into a waiting queue, a confirmation is sent to the sender, and once a confirmation is received the message is moved into the in-box and the email address of the sender is added to the white list; thereafter, a confirmation from that sender will not be required.

There are a few specific issues that the software deals with. In order to catch spammers who impersonate the owner, a "mailkey" can be added to one's own mail. To avoid mail loops, the software keeps the last few emails (the number is variable) in a FIFO queue and checks whether a message occurs more than once. It is also possible to send commands to configure the queue remotely. A limitation of the software is dealing with bounces: The software cannot distinguish between real and spam bounces without accessing the MTA. Another issue that needs to be addressed is that simple challenges can be defeated with a smart auto-responder.

Learning Spam: Simple Techniques for Freely Available Software

Bart Massey, Raya Budrevich, Scott Long, Mick Thomure, Portland State University

Today there are many approaches to the problem of spam, including black/white lists, laws, and automated filtering, which includes feature recognition and classification; these methods can all work together. Feature detection involves extracting information from the header and body of the message. For example, SpamAssassin uses hand-coded features with linear weights combined with threshold values. More sophisticated techniques include information gain and clustering. The simplest format of features is binary, works on all algorithms, and is robust. There are a wide variety of machine-learning techniques. The authors chose techniques that were simple, popular, and educational. The general description of machine learning involves gathering a large number of training instances, measuring statistical properties with respect to a feature set, classifying target instances, and, finally, measuring the accuracy of the classification.

Bart Massey described five different techniques for mail classification, focusing on only a few. The most simple is the minimal Hamming Distance Voting. Here a message is compared against the messages in the corpus to determine its classification; the classification time for this method is huge and it is necessary to have a large variety in the data. The simplest neural net is a single neuron call perceptron, but a more complex neural net works even better. Basically, features are given as input and the network's output provides the classification. There is a need for a representative data set. Tests were run on different corpora, personal messages, and synthetic data. There were enough different characteristics to make a difference. The complex neural network seemed to have the best results, with <1% false positives and 1–3% false negatives. Even humans cannot achieve 0% misclassification rates. Machine learning is a key aspect of the filtering.

NETWORK PROTOCOLS

Summarized by Benjamin A. Schmit

Network Programming for the Rest of Us

Glyph Lefkowitz, Twisted Matrix Labs; Itamar Shtull-Trauring, Zoteca

Itamar Shtull-Trauring started the first talk by comparing network programming to driving a car: An automatic transmission frees the driver from having to make low-level choices, at the possible cost of some performance. Similarly, programmers who want to do networking without caring for peak performance should use a networking toolkit.

The main design goal for the Twisted networking framework was providing a cross-platform solution that still allows access to platform-specific features. It is written in Python, a high-level language that helps beginners avoid common programming errors by providing, for example, automatic boundary checking. The framework is built around an event loop and allows programmers to easily add even complex services such as a Web server to their programs.

In order to show how the Twisted framework can help a programmer speed up development, Conch, an SSH application, was implemented. The implementation contains 5000 lines of code written by a single person, as compared to 64,000 lines by 84 people in the case of OpenSSH. The framework is available for download from *http://www.twistedmatrix.com/* and was released under the LGPL.

In-Place Rsync: File Synchronization for Mobile and Wireless Devices

David Rasch and Randal Burns, Johns Hopkins University

David Rasch identified the space requirement of rsync as a major problem for mobile and wireless devices with little storage capacity. Without this restriction, rsync would be a good tool for synchronization with mobile devices. With the approach presented, the update of a file can be done in place, without the creation of a temporary work copy.

A problem that needed to be solved, however, was that data necessary for later rsync commands should not be overwritten by earlier ones (as he explained, rsync does its work by trying to find blocks from the old file somewhere in the new file). The solution here is the creation of a dependency graph; though it might contain cycles, these are broken up by retransmitting data overwritten by earlier commands, for which two algorithms are implemented.

The benchmarks, which update files typically used on handheld computers, show that there is very little difference in performance as compared to the original rsync implementation. The new algorithm, however, needs more main memory for calculating the dependency graph. This problem can be solved by introducing windowing, which is not yet implemented.

NFS Tricks and Benchmarking Traps

Daniel Ellard and Margo Seltzer, Harvard University

The initial research goal of this paper was to improve the performance of an NFS server by optimizing its read-ahead caching strategy. Daniel Ellard pointed out that 5–10% of the NFS requests arrive at the server in the wrong order, which could hinder it from doing read-ahead properly.

The benchmarks designed to measure performance improvements showed an unusually large amount of variance. This was caused mainly by the fact that modern hard disks have different data transmission rates at different physical areas. New benchmarks that took this into account still showed problems, this time located within the FreeBSD operating system, where a hashtable had been designed too small for today's requirements.

After these problems were solved, the authors improved the NFS server performance by introducing cursors, which made the server able to recognize several concurrent file reads. On a heavily loaded NFS server, these optimizations speed up file requests by a factor of up to 2.4.

BIOS AND VIRTUAL DEVICES

Summarized by Shashi Guruprasad

Flexibility in ROM: A Stackable Open Source BIOS

Adam Agnew, Adam Sulmicki, William Arbaugh, University of Maryland at College Park; Ronald Minnich, Los Alamos National Labs

This won the Best Student Paper award. Adam Agnew presented the first open source PC BIOS that could boot any modern OS. They leveraged earlier work in Linux BIOS, which could only boot Linux. Unlike Linux, some of the OSes rely on services such as video, hard drive, memory sizing, and a PCI table provided by legacy BIOSes. Therefore Linux BIOS could not directly boot these OSes. Legacy BIOSes do a poor job of setting up a PC, requiring modern OSes such as Linux to redo some of the tasks, increasing bootstrap latency. The advantages of using Linux BIOS are numerous: (1) dramatic increase in boot speed – for example, they have achieved a record of a three-second boot – the main bottleneck is the time required to bring the IDE hard drive spin to normal operational speed; (2) small size – an image size of 36KB uncompressed offers scope for more powerful tools to be part of BIOS; (3) open source – it's easily debuggable and royalty free and saves motherboard manufacturers $3–$5 per PC in royalties; (4) remote administration is possible via console support.

The proposed solution employs a combination of Linux BIOS and Bochs x86 emulator, using a custom wrapper functionality known as Adhesive Loader (ADLO), whixh doesn't require the Bochs emulator to be modified. The Bochs emulator has excellent PC BIOS interrupt support, and using an existing mechanism avoided maintaining more software. Together, these provide the missing legacy BIOS support except for the Video BIOS, which is extracted from the Video ROM and packaged along with the above to complete the BIOS. One of the advantages of such a solution is that different components can be stacked, leading to different configurations, of bootloaders, for example, or more sophisticated tools in the BIOS, such as console support.

Possible future work was discussed, which included TCPA, console over other devices such as USB, authenticated booting, virtual machine/virtual machine monitor support in BIOS, transparent encrypted storage, and transparent backup and intrusion detection that cannot be turned off. There was also a demonstration of a quick-booting Windows 2000 OS. During the Q&A, someone asked about menu-based configuration support. The answer was that currently no configuration support is present and it is necessary to re-flash.

URLs:
http://www.missl.cs.umd.edu/
http://Linuxbios.org/
http://bochs.sourceforge.net/

Console over Ethernet

Mike Kistler, Eric van Hensbergen, and Freeman Rawson, IBM Austin Research Laboratory

Eric van Hensbergen talked about console support over an Ethernet device. Traditional forms of console support are through the serial devices that are aggregated through KVM switches in clusters. The main problem with serial console is in supporting denser clusters that have more machines per rack where it is necessary to support only the most essential of the I/O interfaces for reducing both space requirements and the heat dissipation of each machine. An example of such a cluster that researchers in the IBM Austin lab built is known as a superdense server with 200–300 x86 servers in a 42U rack. These servers already support an Ethernet interface and have no room for serial ports on the board.

Two different solutions for console over Ethernet were developed. The first solution, which is a TCP/IP-based Linux console named "etherconsole," uses a kernel interface to the socket library to send and receive console messages instead of performing low-level device operations on the serial device. A major downside of this approach is that console support is possible only after TCP/IP initialization. Problems that occur before this phase will not be reported on the console and, therefore, debugging such problems is difficult. A second solution involves the use of link layer networking, that is, the sending of console messages using raw Ethernet frames with a special Ethernet type and a broadcast Ethernet address. At the receiving end, a program captures these frames and provides the console output. A combined approach was then used for console support: the link-layer approach until DHCP is performed, and the TCP/IP approach afterwards.

Security issues were discussed. The possibility of break-ins and denial-of-service is increased when the console is available over the network. The suggested solutions included a separate private network for console and switch VLAN support. The latter does not address denial-of-service. This was integrated with Linux BIOS into custom firmware at IBM and allowed full system administration and debugging. Some future areas of research in this area were better BIOS integration, serial port emulation, and frame-buffer emulation for OSes that do not support character-based consoles.

Implementing Clonable Network Stacks in the FreeBSD Kernel

Marko Zec, University of Zagreb

Virtual hosting, network simulation, and advanced VPN provisioning require support for multiple protocol stacks on the same physical host. This paper focuses on the design, implementation, and performance of an experimental clonable network stack in the FreeBSD kernel. By separate stacks, the author means independent states – routing tables, interfaces, and firewall rules – rather than independent protocol code. In other words, this work does not support the creation of a separate protocol stack that is not already supported by the OS. Virtualizing a network stack for the above applications is a more lightweight alternative to a traditional virtual machine.

The key design goals were: API/ABI (Application Binary Interface) compatibility and low or negligible performance overhead. A naïve approach to implement clonable stacks is to add an array dimension to the kernel networking data structures. Such an approach, however, increases the amount of kernel code modification. Instead, the approach used was to group the kernel data structures that maintain the state of the network stack together and provide access through an additional level of indirection. This grouping is termed a "virtual image" and has an instance of network stack associated with it. Virtual images are organized in a hierarchy similar to the UNIX process hierarchy that helps in binding unmodified programs to different network stacks.

There was also a discussion on the implementation of CPU load and usage accounting/limiting per virtual image, which helps in resource management and avoids receive livelock. The performance degradation is hardly noticeable, around 3.5% lower than an unmodified kernel where the test system had one active network stack among 128 stack instances. In some tests, the performance improved slightly (5.7%) due to better locality of kernel network stack variables which led to higher CPU cache hits.

URL: *http://www.tel.fer.hr/zec/BSD/vimage/index.html*

FILE SYSTEMS

Summarized by Manish Prasad

StarFish: Highly Available Block Storage

Eran Gabber, Jeff Fellin, Michael Flaster, Fengrui Gu, Bruce Hillyer, Wee Teck Ng, Banu Özden, and Elizabeth Shriver, Lucent Technologies, Bell Labs

Michael Flaster presented this FREENIX award paper. The principal contribution was the dissemination of a replicated block storage system to the open source community, built from commodity servers running FreeBSD, connected by standard high-speed IP networking gear. StarFish is not a distributed file system. It assumes a single-owner scenario and thus doesn't deal with issues resulting from multiple writers.

StarFish architecture comprises a commodity server with an appropriate SCSI or FC controller, called host element (HE), which helps the client host access data from a set of storage elements (SEs) that talk to the HE using TCP/IP. An SE forms a single unit of replication. The SEs could be connected to the HE via either a dedicated link or the Internet. Writes to StarFish are considered to be complete on receiving ACKs from all the SEs that form the required quorum.

The presenter projected steadfast reliability as the unique selling point of StarFish and presented in detail how it behaves when things go wrong, like restarting an out-of-date SE or the failure of an HE (manual failover), and also how the HE ensures that a quorum of SEs are in sync when one SE happens to be slower than the other.

Finally, the author presented some interesting experimental results for read and write availability against various quorum sizes and arrived at a quorum size of two and a set of three replicating SEs as a recommended configuration to achieve good read and write availability. He concluded that SEs not in quorum could be connected over the Internet, thus eliminating the need for a dedicated high-bandwidth low-latency link.

Secure and Flexible Global File Sharing

Stefan Miltchev, Jonathan M. Smith, Sotiris Ioannidis, University of Pennsylvania; Vassilis Prevelakis, Drexel University; John Ioannidis, AT&T Labs – Research; Angelos D. Keromytis, Columbia University

Stefan Miltchev presented this work on authentication in network file systems. The paper presents Distributed Credential FileSystem (DisCFS), which uses trust management credentials to "directly authorize actions rather than divide the authorization task into authentication and access control" and "to identify files being stored; users; and conditions under which their file access is allowed." DisCFS is intended to address the weaknesses of existing file access control mechanisms, which are either too coarse-grained (e.g., Web, FTP) or are unsuitable for use across administrative domains (e.g., NFS).

DisCFS uses a direct binding between a public key and a set of authorizations. A user can delegate trust to another, which results in creation of a chain of trust, similar to systems like SPKI. Only a subset of privileges may be delegated to another user, making privilege escalation impossible. DisCFS can be implemented over any existing mechanism of data exchange (NFS, HTTP, FTP) and can leverage existing IPSec infrastructure for secure client-file server communication.

The experimental platform comprised a set of machines with modest hardware running OpenBSD. Measurements using Bonnie micro-benchmark showed that write throughput was comparable to that of NFSv2, although read throughput was observed to be lower than NFS. The Postmark benchmark, representing a heavy workload of small files, was used for versions of DisCFS without any security and with and without credential caching.

The talk concluded with a discussion of future work, the most noteworthy concerning implementation of access control beyond permission bits and avoiding the use of inode numbers as file handles because of security holes created by inode number reuse.

The CryptoGraphic Disk Driver

Roland C. Dowdeswell, NetBSD Project; John Ioannidis, AT&T Labs – Research

Dowdeswell presented the CryptoGraphic Disk Driver (CGD), "a pseudo-device driver that sits below the buffer cache and provides an encrypted view of an underlying raw partition." CGD targets the problems arising from laptops and other portables "growing legs" and vanishing from public places! Here, "protection of data from other concurrent users is not essential, but protection against loss or theft is important."

Due to its positioning in the I/O stack (just above the disk driver), it is completely transparent to file systems. Some of the goals of the work are to maintain good performance while providing adequate security, ease of use, and seamless integration into the OS release. The in-kernel driver supports an ioctl interface to attach CGD to an underlying disk device or partition and configure the required parameters such as encryption algorithm, key length, IV method, etc. They define a modular framework for adding cryptographic algorithms.

The driver was evaluated on DEC Personal Workstation 500a, Pentium 4–based PC, and an IBM Thinkpad 600E. Results were presented for read-and-write throughput for various block sizes, comparing various encryption algorithms – Blowfish, AES, and 3DES – against unencrypted I/O. As expected, encrypted I/O throughput edges closer to raw I/O with the increase in block size. The talk concluded with a discussion of future work, which included addressing the problem of key revocation and leveraging hardware cryptographic accelerators to achieve better performance.

X WINDOW SYSTEM

Summarized by James Nugent

Xstroke: Full-Screen Gesture Recognition for X

Carl D. Worth, University of Southern California

The goal of Xstroke is to provide full-screen gesture recognition for X Windows. A gesture is recognized and is passed to the system as a keystroke or a series of keystrokes. The focus is on handheld devices, so it is important that the entire screen can be used for recognition, unlike some systems that have a special "gesture area." The size and location of the gesture are also important. Finally, the gesture "draws" on the screen with a transparent color, thus allowing the gesture to be seen as it is made, but not to obscure data beneath.

One difficulty is toggling gesture recognition on and off. Several approaches were discussed, but none was ideal. The Recognizer is a 3x3 grid based on the bounding box of the stroke. A grid has the problem that similar strokes may look different near corners; thus regular expressions were used to recognize the strings of grid numbers that defined a stroke. An additional problem is that if the device is held tilted, strokes will also be tilted. The solution to this is that the common horizontal strokes (backspace and space) were used to reorient the grid when one was recognized. One nice feature is that tilting the device enough to cause incorrect recognition produces garbage characters, alerting the user to correct the problem with backspace. More information and software are available at *http://www.xstroke.org*.

Matchbox: Window Management Not for the Desktop

Matthew Allum, OpenedHand Ltd.

Matchbox is an X Window manager designed for small devices that have low memory, no keyboard, and limited CPU power. Matchbox is designed to be small, fast, flexible, and configurable. It has some restrictions specific to its environment: Only a single full-screen app is supported at a time, and applications cannot resize windows or make windows larger than the screen. A toolbar/virtual keyboard is always visible.

Matchbox's focus is on providing window management that is also compliant with standards, most notably the ICCCM standards. Matchbox is themeable, has runtime and compile-time configuration options, and is extensible via plugins. It also has a PDA-style app launcher.

X Window System Network Performance

Keith Packard and James Gettys, Cambridge Research Laboratory, HP Labs

Several X clients were set up to talk to an X server attached to a passive packet-monitoring tool and then to low-bandwidth X (LBX) and SSH proxies with a NISTNet router. (NISTNet is capable of producing a variety of delay/bandwidth-limited network conditions.) The xplot performance visualization tool was used to understand the raw packet data. The usefulness of this tool is difficult to overemphasize; it allows the network trace to be examined in detail and problem areas to be picked out easily.

The LBX proxy uses application-specific compression to reduce the size of requests. In general, LBX was found to be of very little utility in a modern setting: some of the requests it can compress are obsolete, plus bandwidth for X requests is now a problem only for large image files. SSH's gzip compression, in fact, was found to be superior in almost all cases.

In general, latency dominates bandwidth, and most of the latency comes from synchronous requests. Many of these can be eliminated by batching them, as with internAtom requests (already done by some toolkits). Finally, restructuring applications can reduce the effects of latency. Image files were found to be the dominant reason for bandwidth usage. It was suggested that using original image formats, which are usually compressed, would be helpful.

Client-side fonts were also studied; they had the effect of significantly reducing the round trips and, hence, application startup time. Bandwidth usage is about the same, although compressing the glyphs for transport would reduce this.

EXPERIENCES

Summarized by Raya Budrevich

Building a Wireless Community Network in the Netherlands

Rudi van Drunen, Dirk-Willem van Gulik, Jasper Koolhaas, Huub Schuurmans, and Marten Vijn, Wireless Leiden Foundation

The purpose is to provide free wireless access in historical cities in the Netherlands. The technology uses 802.11b with three available concurrent channels without interference. The network is on ISM band, shared with ham radio, microwave, and vehicle ID. It is an IPv4 network with a private address space, routing by OSPF, with ISC-DHCP and ISC-DNS; the network is transparent to the user and provides no user-level security.

Currently, there is a 20km^2 outdoor coverage for the city of Leiden, 20+ nodes, 300 private users, and three proxies to connect to the Internet; many local schools and libraries are also connected. There is a need to find free locations for the antennas, since there is no funding to pay rental fees. At every site the strength of the other nodes and the interference are measured. Network simulation software is used to determine the construction of all sites.

A node consists of multiple antennas with donated PCs or embedded systems. The systems run FreeBSD because it is stable, single distribution, and tagged release. Industry-standard wireless cards are used.

The wireless network is available freely to all inhabitants of Leiden. The Wireless Leiden Foundation is a nonprofit organization that tries to create strong connections with schools and universities. The network has many uses, such as P2P, gaming, and VPN. On any given day, about 100 users are connected.

Community acceptance, project management, and interference were the main problems facing the project.

OPENCM: EARLY EXPERIENCES AND LESSONS LEARNED

Jonathan S. Shapiro, John Vanderburgh, and Jack Lloyd, Johns Hopkins University

OpenCM is a new configuration-management system that uses cryptographic authentication and high integrity. The architecture uses cryptographic naming, and the content of a configuration file is a DAG; every revision is a pointer to the root of the DAG. If we use crypto hashes for the pointers and sign the revision records, we get end-to-end integrity and auditability. SSL is used as the authentication technology. Because of the permissions setup of the system, anonymous access is easy and convenient.

Currently authentication and integrity work well. During the last two years there have been only three breaches. The hazards include OpenSSL bugs yielding an unreliable transport. Boehm-GC flaws yield leaked memory and regular server hangs.

Several decisions, during implementation, that seemed plausible led to errors later on:

1. Using a texty format for debugging – This cost 30% more space and didn't compress, and file systems didn't respect cases. Therefore one should plan ahead for binary format.

2. Server-side integrity checks – Files might be damaged when they reach the server. The server needs to serialize/deserialize, which leads to the server knowing the object schema.

3. Build a simple file-based store first – One file per object was stored, using gzip to compress, but in switching to binary format, it was discovered that 60% of the files were less than 500 bytes.

4. Build an event-driven server – Used non-blocking I/O and an event loop, which simplified the code and the server but caused problems with SSL.

5. Use GC and exception handling – This was well engineered for memory management but has many platform dependencies, leaks memory in large objects, and doesn't work on OpenSSL.

To fix GC and exceptions a new management, GC_SCOPEs, is introduced. Currently, schema issues described in the paper are all fixed in the development branch. There's still a need to modify storage format and replace Boehm-GC with a portable application-specific collection strategy.

FREE SOFTWARE AND HIGH-POWER ROCKETRY: THE PORTLAND STATE AEROSPACE SOCIETY

James Perkins, Andrew Greenberg, Jamey Sharp, David Cassard, and Bart Massey, Portland State University

The research group consists of undergraduate and graduate students in science and engineering at Portland State University, people in local industry, and local aerospace enthusiasts. The current model reached 3.6km; the next-generation model will leave the atmosphere. The goal is to put nanosatellites in orbit.

The active guidance computer on the rocket determines the rocket's current position, heading, and course; the computer then steers the rocket to keep it on course. In order to determine this information the computer uses several sensors: GPS, inertial measurement unit (IMU), magnetometers, and pressure and optical sensors.

The launch tower is used to launch the rocket, view rocket status, and send emergency commands to the rocket. The on-board system includes the flight software avionics firmware.

It was difficult to decide which embedded operating system to use. There were several candidates but RedHat's eCos was selected. It has all of the needed criteria: RT, POSIX, free for use, small.

Commercial OEM GPS boards don't work because of their software limitations in determining acceleration velocity and altitude. GPL-licensed firmware for GPS receivers uses eCos. Differential GPS base station and receiver provide precision timing and altitude determinations.

Future work includes creating an enhanced inertial measurement unit, developing next-generation navigation algorithms, investigating loose coupling of IMU and GPS, GPS aiding of the IMU unit, deep coupling of IMU/GPS and other sensors, and adding a steerable hybrid motor.

PRIVILEGE MANAGEMENT

Summarized by Benjamin A. Schmit

POSIX Access Control Lists on Linux

Andreas Gruenbacher, SuSE Linux AG

The POSIX.1 access permission model is not always sufficient, especially when interacting with Windows clients via SAMBA. The abandoned standard POSIX.1e (the last draft, 17, is available to the public) is taken as the basis of most ACL implementations.

In the Linux implementation, which has become an official part of the 2.5 kernel version, access permissions (read, write, and execute) can be granted to the owner, named users, the owning group, named groups, and others. A mask is used to retain backward compatibility for non-ACL-aware applications. Default permissions make it possible to inherit permissions that were set at directory level.

The Linux ACL implementation can be used for ACL-aware network protocols, the most important ones being NFS (only version 4) and CIFS (formerly SMB). Since these two protocols implement ACLs in a different way, a mapping needs to be done here. The Linux implementation does not support granting somebody other than the owner the ability to change permissions.

Privman: A Library for Partitioning Applications

Douglas Kilpatrick, Network Associates Laboratories

In this talk, Douglas Kilpatrick described the Privman privilege management library. Some UNIX applications make use of privileges that normal users can't acquire and, therefore, run with root privileges because UNIX does not yet support relinquishing access privileges. This is no problem as long as the applications do not contain any bugs (which could lead to root privileges for any local user).

Privman is implemented as a user-space library which mirrors some calls to the libc as well as some system calls. A program linked to the library starts by forking into a privileged server and a client that gives up all its access privileges. When the client needs to make a privileged call, it contacts the server on a pipe. The server than decides, based on a policy file, whether the access should be granted.

The main advantage to Privman is that few changes need to be added to the source code. The overhead induced by the wrapper calls can be huge for a single system call, but because these calls occur rarely, the performance of a typical application decreases by only about 5%. Privman has been used to adapt OpenSSH and wu-ftpd for privilege management.

The TrustedBSD MAC Framework: Extensible Kernel Access Control for FreeBSD 5.0

Robert Watson, Wayne Morrison, and Chris Vance, Network Associates Laboratories; Brian Feldman, FreeBSD Project

In his dense talk, Robert Watson explained the implementation of the Mandatory Access Control system within FreeBSD 5.x. The MAC framework is implemented partly within the kernel, partly in user-space. Common applications require more than just the standard UNIX security mechanisms. Operating system support for access control is important; without it, applications would have to cope with several different security mechanisms, the result being mostly an intersection of the individual permissions.

For MAC, the FreeBSD kernel has been extended by additional events. MAC-aware applications register their interest in some events, then get these events, and possibly de-register afterward. Access policies are divided into adaptations of traditional access policies, traditional MAC policies, and new ones like a port of the SELinux FLASK policy to FreeBSD.

The main benefit of MAC is that the source code does not have to be modified to use it. The implementation still lacks support for multi-threaded applications and multi-threaded kernel threads. Also, object labeling within the kernel currently decreases performance, which could be solved by introducing a cache.

KERNEL

Summarized by Francis Manoj David

Using Read-Copy-Update Techniques for System V IPC in the Linux 2.5 Kernel

Andrea Arcangeli, SuSE; Mingming Cao, Paul McKenney, and Dipankar Sarma, IBM

Locking and synchronization are extremely important tools in multi-threaded environments. Atomic increment, the key to locking, is possible on current CPUs. However, newer processors (like the Pentium 4) consume significantly more cycles than older processors during a single atomic increment. This is because the whole pipeline needs to be flushed, and this is an expensive operation. For data that is read-only, paying this penalty is unreasonable.

Read-Copy-Update (RCU) is a technique that "allows lock-free read-only access to data structures that are concurrently modified on SMP systems." To illustrate RCU concepts, let us look at an example. In a linked list, deletion of an element and traversal of the list cannot happen simultaneously. This is because the traversal code might reference a pointer that is freed by the deletion code. The RCU-based solution is to defer the freeing of the pointer until it is certain that no other code is traversing the list. The paper references a couple of more efficient solutions to this problem.

The authors have successfully applied RCU techniques for semaphore data structures in the Linux 2.5 kernel. There is a dynamic array called ipc_ids that needs to be traversed for all semaphore operations. This array is protected by a global lock. This design prevents semaphore operations from proceeding in parallel. Using RCU, the global lock was eliminated and deferred deletion of semaphores was implemented. The total number of new lines of code added to the semaphore implementation was only 151. Micro-benchmarks show very good results. In the future, the authors plan to use RCU to optimize more code in the kernel. The current code is available under the GPL license.

URLs:
http://www.rdrop.com/users/paulmck
http://sourceforge.net/projects/lse

An Implementation of User-Level Restartable Atomic Sequences on the NetBSD Operating System

Gregory McGarry

A Restartable Atomic Sequence (RAS) is a mechanism used to efficiently implement atomic operations on uniprocessor systems. It was designed to provide atomic operations on processors that don't provide them. On processors that do provide them, RAS increases processor performance.

System calls are one option when it comes to performing atomic operations for user threads. This emulates memory-interlocked instructions. However, this comes with a large overhead. RAS is a better solution to this problem. An RAS is basically some code that provides some primitive like test and set or increment a variable (e.g., `count++` in C). In order to guarantee that this code is called atomically, a user thread should not be preempted in the middle of this code. If it does get preempted, then the code needs to be restarted. This restarting would ensure that the operation is performed atomically.

RAS has been implemented for the NetBSD OS. The user thread registers the entry and exit points of the RAS with the kernel. Whenever there is a context switch, the kernel checks to see whether the execution was in the middle of the RAS, and if it was, the RAS is restarted when the thread returns. Thus, atomic operations are provided. Also, one cannot write arbitrary code for RAS and expect things to work correctly. An RAS should have the following properties. It should have a single entry and exit point, should not modify shared data, and should not invoke any functions. This ensures that, on restarting the sequence, the thread is restored to a correct state. Thus, the responsibility for getting things to work correctly is shared between the kernel and the user thread.

The user interface to the system is similar to that provided by mmap. RASes are registered with the kernel. The kernel checks for such sequences in the `cpu_switch` function and restarts sequences if necessary. Performance studies show that this approach is better than the syscall approach. RAS is currently transparently used in the NetBSD pthread library for uniprocessor machines.

Providing a Linux API on the Scalable K42 Kernel

Jonathan Appavoo, University of Toronto; Marc Auslander, Dilma Da Silva, David Edelsohn, Orran Krieger, Michal Ostrowski, Bryan Rosenburg, Robert W. Wisniewski, and Jimi Xenidis, IBM T.J. Watson Research Center

K42 is a new research OS kernel designed to be highly scalable and extensible and written in object-oriented style. It also supports online reconfiguration of kernel services. K42 pushes a lot of usual kernel functionality into user-space, enabling efficient IPC. Also, global data and locks are avoided in order to improve performance.

The motivation for this work was to run all standard Linux programs on K42, thus increasing the number of applications available to those working with K42. This paper describes the challenges that the authors faced in providing the Linux API on K42. Linux processes were supported by mapping them to K42 processes. A K42 process called the `ProcessLinuxServer` maintains these relationships. Fork was implemented by placing most of the code in user-space. The file descriptor table was lazily replicated for performance reasons. For signals, all state was kept in the client. To provide a Linux environment, a new version of glibc, targeted toward K42, was compiled. `Exec` was also implemented in user-space. This involves an unload operation followed by a reload operation. Namespace resolution for files was also implemented in user-space. The entire Linux emulation layer was provided by a modified Linux kernel, with K42 as the target architecture. Currently, the project has a fully functional implementation for 64-bit architectures and is available for download under the LGPL license.

URL: *http://www.research.ibm.com/K42/*

15th Annual Computer Security Incident Handling (FIRST) Conference

Ottawa, Canada
June 22–27, 2003

Summarized by Anne Bennett

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

TUTORIALS

Network Forensics

E. Larry Lidz, University of Chicago

Looking at problems from the network: Often, we don't have access to the machine we suspect of being involved in a compromise (the machine is physically inaccessible, it belongs to a student, etc.). Network audit logs can save the day: If we are able to go back and show where a problem came from, we can quickly resolve the problem. Also, if it is necessary to turn over evidence to authorities, it can be legally easy to turn over network audit logs, which tend not to contain confidential information, whereas it can be really tricky to turn over the hard drives of compromised machines (because confidential information must be protected).

The speaker gave a crash course on TCP/IP, connection establishment and termination, IP addresses and network masks, and well-known ports.

Why use network audit logs if we already have one or more of an IDS (Intrusion Detection System), the ability to do system forensics later, the ability to sniff traffic as needed, or firewall logs?

- IDS: IDS logs only known suspicious traffic, but we often need complete logs. Also, some kinds of suspicious packets cannot be logged: Out-of-sequence FINs can "turn off" IDS monitoring for a connection, and too many fragments can overwhelm an IDS.
- System forensics: A compromised system has often had its data altered, attempts to fix the problem may have destroyed data, and the intruder may not have written anything to disk.
- Sniffers: Often it is too late to turn on the sniffer – the problem has already happened. Also, because the entire packet is logged instead of just the header info, disk consumption is massive and there are more serious privacy issues.
- Firewalls: Usually they do not contain as much information as audit logs (except in the special case of application firewalls).

All of the above have their place, but it is still useful to have network audit logs to help investigate break-ins, to determine how our network is being used for both legitimate and illegitimate purposes, and to get a picture of "normal" use of the network. Also, the network audit logs are (or should be!) on a trusted, hardened system, so their information is quite reliable.

Well-known audit log systems:

- Cisco NetFlow logs straight from Cisco (and some other vendors') routers and switches (can impact performance). These logs are easy to search.
- Argus logs from the spanning port of a switch and is able to log packet contents if desired. Argus is a free product, and there is a more DoS-resistant and featureful related commercial product named Gargoyle.

Active probing (port scanning), though useful during investigations, may tip off an intruder that you have noticed them. Make sure you have the authority to scan any machine you want to probe.

Some useful tools are:

- nmap: finds open ports, O/S fingerprinting.
- nc: text communication with tcp port.
- pnscan: a massively parallel scanner, can grab banners, even pass text to the port, but can sometimes miss hosts because of timeout problems.
- rpcinfo: identifies RPC services running.
- smbclient: NetBIOS info, comes with Samba.
- dcetest: like rpcinfo but for DCE services.
- ibnet: can craft specific packets.

It is extremely important to keep system logs in a trusted place (central log server) and watch them for unusual activity.

We were shown examples of how NetFlow and Argus are used to investigate incidents. For example, a bunch of Solaris hosts were reported to be compromised. The investigators picked a relatively "quiet" one (fewer logs to look through) and found out that something had scanned it for port 111 and connected to ttdbserver. Then another host had connected to port 1524 (which was later determined to be a back-door port) and rsh'd to a machine on the same network as the original "scanner" (this was interpreted to mean that the victim was downloading a rootkit); then IRC traffic started up.

This analysis suggests that to find additional compromised machines, there are a few things that can be looked for: successful connections to port 1524 locally, local machines rshing to the same offsite machine, and, possibly, IRC traffic (unless there is lots of legitimate IRC traffic). Ideally, you should cross-correlate multiple sources for better reliability, and, of course, hand-verify hosts likely to have been compromised.

Hints on what to look for in network logs when investigating an incident:

- When was the first traffic to a back door (first successful connection to a port which previously had no traffic)?
- When did IRC traffic start?
- Was there a sudden increase in traffic on a given machine?
- Did unexpected traffic start shortly after a connection to a service running on the machine?
- Once a back-door port has been found, what machines connected to it?

These techniques proved useful for Slammer Worm cases (where the symptoms were that the external net died and several internal routers crashed; Cisco reported a network-wide DDoS) and in cases of a specific compromise. The initial Slammer analysis and identification of an initial set of Slammer-compromised machines was done in 15 minutes!

Note: There are now tools (e.g., softflow) to send flow logs off from UNIX machines which are acting as routers.

Secure Coding: Principles and Practices

Kenneth R. van Wyk, Tekmark Technology Risk Management, USA; Mark G. Graff

The intention of the authors of the O'Reilly book of the same name was not to give tons of source code examples but, rather, to give programmers the ability to think clearly about secure coding. When we build bridges, we don't make trucks drive over it, see if it collapses, then if it does, move on to "bridge v.1.1." We need a similar perspective for the construction of software.

Van Wyk strongly emphasized a mindset that needs to be adopted to have any chance of creating reasonably secure software. Try to look at your software from the point of view of an attacker: how can it be subverted? With that in mind, it is possible to apply a set of architectural principles and make sure that the proposed design respects those principles, such as: restrict privileges granted, assign responsibility to individuals (make sure that actions can be traced back to their originators), log sufficiently well that events can be reconstructed, and fail safely.

We were given examples of risk-assessment questions and broad categories of risk mitigation strategies, including avoiding the risk by removing the cause or the consequence, limiting the risk by detecting and responding to problems, transferring the risk to another party (e.g., by buying insurance), and assuming the risk (being prepared to deal with the consequences of a problem).

When maintaining or modifying existing software, be aware of the design intent and the security model of the original, otherwise you risk introducing problems you have no idea about.

Near the end of the presentation, we were given some specific coding tips. The fairly low emphasis on this issue in the presentation maps well onto the amount of time that should be devoted to the implementation of a project, versus planning and design, which should receive the lion's share of time. Most of those coding tips ought to be well-known by now!

Finally, don't neglect the environment in which the code will be used: Secure the OS and network, monitor logs, keep up to date with patches, and so on. And, of course, don't forget to test every component and test at every stage: Your environment changes. Automate your testing.

Creating a National Alerting and Reporting Service

Nienke van den Berg, GOVCERT.NL, the Netherlands; Jeffrey Carpenter, CERT/CC, Carnegie Mellon University, USA; Graham Ingram, AusCERT, Australia

The Internet is no longer just an academic and research network, but has become part of a national social and economic infrastructure. Governments are starting to look to national CSIRTs to help keep the Net viable as such an infrastructure. Entering a "national picture" has raised legal, process, alerting (info provided to the public), and reporting (info received from the public) issues for existing (and new) CSIRTs.

In deciding when to issue an alert, you should evaluate the particular vulnerability or event based on its technical impact (a.k.a. objective impact – the probability of technical and economic damage) and on its social impact (a.k.a. subjective impact – the perception of safety, influenced by the likelihood of media exposure).

How to provide alerts: Web (need a good content management system), email (lists can get quite large), mass media, and SMS.

Not all of these are used for each alert: There is a decision matrix that maps technical and social impacts onto a "media mix" (set of channels through which the alert is sent). A communications advisor is used to ensure that alerts are "not too technical" when they are aimed at the general public.

Email alerts are PGP-signed, of course, but since the general public may not be in a position to check the signature, it helps to have a very well–publicized Web site address to reduce the chances that an impostor could hijack (forge) the distribution channel.

Here are some questions to ask in deciding what is worthy of advisories: What is the current threat posed by this vulnerability? How bad would it be if the vulnerability were publicly known? if the vulnerability were publicly known and being exploited? How bad is the incident already? How much worse can it get?

CERT/CC has developed formal metrics to rank vulnerabilities according to the severity of the impact on a "typical" site; these metrics involve questions with numerical answers by which a weighted total is computed. This metric is used to help decide whether the problem justifies an email advisory or whether it is sufficient to post the info to the Web site. The metric is also used to decide which issues to work on first. The questions involve the ease of exploitation of the vulnerability, how widely known the vulnerability is, the risk to the Internet infrastructure, and how many systems would be affected and the severity of this impact.

CERT/CC has also developed formal metrics to rank incidents (as opposed to vulnerabilities). With respect to incident activity relating to a particular vulnerability, CERT/CC differentiates between current activity and potential additional activity, not counting what has happened so far (exploitation of a vulnerability), in order to determine the goals of any action.

CERT/CC also adds a "uniqueness factor" so that they don't issue multiple advisories or incident notes on the same subject, even if related-incident activity continues.

To evaluate current impact, questions are asked about how many people and/or machines are affected, at how many unique sites, what the importance of the systems affected is, what the impact during and after the actual activity is, and how complicated the attack method is.

To evaluate potential (additional) impact, questions involve how many people and/or machines are affected by the vulnerability, the importance of the systems, how rapidly the problem is spreading, and how complicated the attack method is (i.e., how many people could write an exploit, therefore how likely the exploit is to appear soon).

To set up an alerting service you need money; an operational CSIRT (center of operations); systems for Web service, Web content management, email list management, and project and office management; and technical, communication, and legal expertise. Legal issues must be addressed (general terms and conditions of service, privacy policy and disclaimers, contracts and service level agreements), PR must be handled, and internal processes must be clear.

The purpose of a reporting service (e.g., AusCERT) is to collect, process, and analyze computer security incident reports and share sanitized aggregate reporting with an appropriate audience. This should provide meaningful intelligence about trends and modus operandi with respect to network attack activity.

The goals of the service are to promote the use of mitigation strategies, raise awareness of computer security issues, keep people up-to-date with threat activity and trends, provide information about attack data which they would otherwise not be able to obtain, and provide value-added analysis of this data.

Any reporting service must be able to protect the data of the reporters (and reassure them that this will indeed be the case). However, a reporting system as a black hole is not a good idea: People are motivated to provide data by having access to sanitized and/or aggregated data and statistics.

It is important to have a clear workflow for the reporting process, not least because it forces you to make sure that your stated goals are in fact being addressed in your processes. Also, privacy considerations dictate that you must have guidelines about what information is communicated to whom and under what conditions. Finally, the process makes clear what resources are needed to do the work.

AusCERT described their complex reporting service system. First of all, it was necessary to collect incident data online; the alternative of a call center staffed by 40 or so people was simply not viable. In terms of security, the last thing a CSIRT wants is to have the press report that the CSIRT suffered an intrusion! It is equally important to keep data segregated, so that one reporter does not have access to data submitted by another reporter. The automated response system must automatically do triage on the incoming data: If something is reported for the first time, it probably requires human attention. The system must assign "threads" to incidents so that correspondence and actions on a particular incident can be correlated.

The types of requirements that must be met in order to set up a reporting service are the same as for an alerting service, but the actual contents are different. Most CSIRTs need to use a Web form to collect data in order to automate the process, including the initial analysis of incoming reports. The system must scale very well. A worm might result in thousands of reports from the general public, for example, but if there is a new and dangerous exploit nestled in among these thousands, the system must quickly pick it out for human attention. It must also be possible to quickly change the Web form to respond to emerging issues; for example, a new type of incident might require the collection of a new category of data.

How to handle anonymous reports? AusCERT accepts and stores anonymous reports, but takes no action unless they can later be correlated with reports from validated sources. Don't forget to try to collect consent from reporters with respect to sharing confidential information with specific bodies when needed. Arrange to refuse or otherwise deal with certain types of reports that are not computer-related or where liability would be an issue – reports of pure criminal activity such as murders, for example, are not wanted in a computer security incident reporting scheme. Reports of cybercrime accompanied by a refusal to release information to law enforcement authorities should also be discouraged or refused.

Alerting and reporting services tips and tricks. Share knowledge and expertise with other CSIRTs; integrate the alerting and reporting activities with the processes of the CSIRT; do alerting first to build credibility for later report collection; keep close contact with target groups to improve the quality of the alerts; start a newsletter service for people who are not specifically interested in alerts; and establish good national press contacts in case escalation is needed.

INTRODUCTION TO ADVISORIES

Andrew Cormack, UKERNA, UK

1. Why do vulnerabilities happen?

- Laws of nature: Because computer networks are complex systems, they will certainly contain errors, some of which will have security implications.
- Customer demands: People ask for computers which are "easy" to use; rarely do they demand computers which are "safe" to use. Therefore, vendors sell systems with everything turned on. When Sun tried the opposite, they received many "non-working" returns! And while users will turn on what they need, they will rarely bother to turn off services they don't need.
- Vendor pressures: Vendors are under economic pressure to ship new features fast, with the result that testing is incomplete. Testing for security (the absence of unintended functionality) is harder than testing for intended functionality.

2. Sources of information about vulnerabilities:

- Incident reports are clearly a reliable indication that there is a problem! However, the information obtained may have been obscured by the attackers and may be hard to interpret.
- Full-disclosure communities (e.g., BugTraq): Often the information is up to date but the quality is variable, and there is more emphasis on problems than on solutions.
- Crackers (via published tools): The information is very current, but malware has to be handled with extreme care, since it may contain additional attacks aside from the attack it claims to be performing. It can be difficult to extract good information from the tools; reverse engineering is required.
- Vendors: Information can be of very good quality, but vendors can be very slow, and, because of competing motives, the information may be incomplete or may understate the impact of the problem.
- Commercial services such as antivirus vendors, ISS, etc.: The quality of the information is generally high, and their advisories usually come out before the vendors' advisories. However, there may be restrictions on distribution, and, again, competing motives may affect the information, for example, by overstating the impact in order to sell their services.
- Other CSIRTs have similar motivations as us and are generally trustworthy. But there may be restrictions on distribution of the information. In addition, CSIRTs may be slow, depending on their policies and on the resources available to them.

Clearly, not all information is equal, so it helps to use multiple sources to ensure speed, reliability, and completeness. You should verify this information by correlating the information from independent sources, testing for yourself, and using the trustworthiness of the source as a criterion.

3. CSIRT tasks with respect to vulnerabilities:

- Planning: Plan in advance how to use information for the benefit of your constituency and minimize harm; there's no point in sending out an advisory that says, "There's a problem but there's nothing you can do."
- Distribution: Simply pass on existing information, perhaps translating to a local language or sending only information that is relevant to a particular constituency.
- Interpretation: This can involve gathering information from multiple sources but, most importantly, interpreting the information to suit it to the skill level or common platforms for your community, and/or adding your own introduction to place the information in context. When writing an advisory, list the important information in the first paragraph: who is vulnerable, whether the problem is exploitable remotely, what the damage is, and the immediacy of the threat. Then discuss how to fix the problem, mentioning any side effects of the fixes. IEEE 1044 describes ways to describe software anomalies in something like these terms.
- Investigation: Be clear about why you are investigating a problem: to better understand the problem?

because you intend to notify the vendor? to check or create patches and workarounds? You can investigate based on incident artifacts (e.g., files left on a compromised machine), source code if available, or test systems (which are not on a public network!).

- Coordination: This refers to working with vendors to resolve a problem. This can be tricky; trust must be built and is easy to lose, and the motivations of the parties may compete. Vendors don't want bad publicity, but our constituency needs patches to prevent incidents, and so do other sites. (But will any publicity of our efforts increase the risks to those other sites?)

The presenter is a partner in the TRANSITS project, a European initiative to provide training on issues related to the provision of CSIRT services, and has documented the mechanics of how to write an advisory. The relevant links are, respectively:

http://www.ist-transits.org/
http://www.ja.net/documents/gn_advisories.pdf

ADVISORIES

Michael Caudill, Cisco Systems Ltd, UK

At Cisco, an advisory usually starts with some kind of notification of "bad news" from an independent researcher or a customer. It is a good idea for the vendor to send some kind of response within 24 hours, because otherwise the person will feel that their message has not been heard/read; Caudill recommended using a PGP-signed reply. Nevertheless, vulnerability reporters should keep in mind that the vendor may be working different office hours, have different national holidays, or be a small shop that has only one person receiving vulnerabilities, so it is reasonable to allow a week or so for the vendor to respond.

Once the vendor has received and acknowledged a vulnerability report, it needs to reproduce and verify the problem: What exactly is the problem? Determine workarounds, and whether they are effective and feasible. Find the fix. Determine whether other vendors are affected. Determine if the problem deserves an advisory, based on ease of exploit, impact, and so on.

The "reproduction and fixing" stage may take from a few hours to a few weeks, depending on how much information the vendor has, the complexity of the setup, and, of course, the difficulty of debugging. Also, teams may be small and working on other cases already.

The advisory is the vendor's official response to notification of the vulnerability. It is best to prepare the advisory before you need it, in case events force you to publish prematurely. Use an informative title, the status (draft, interim, final), the date in GMT, a summary to be read by management and by techies to determine whether the advisory applies to them at all.

Other information that should appear in the advisory: which hardware and software models are and are not affected; specific configurations that are affected if applicable; what causes the problem; the symptoms of the problem (crash, performance slowdown, etc.); the actual consequences (unauthorized access, DoS, information leak, etc.); who discovered the problem (give credit where due); whether the problem is being exploited, including the names of known exploit tools where applicable; links to other advisories; CVE number. Determine what language(s) should be used in the advisory.

Obtaining the actual bug fix from developers can be difficult, because of the commercial pressures to provide features they are under.

If other vendors are affected, either notify them directly (especially if the number is small) or hand off to a CSIRT coordination center of some kind to deal with contacting the other vendors.

When the fix arrives, check it, and do regression testing to make sure that the fix doesn't break anything else (unless the fix is really trivial). Before releasing all this, make sure, if the fix will require a hardware upgrade, that this hardware is available through the normal channels.

Finish off the advisory with information about the migration path and how to obtain fixed hardware or software. Run the early copy past developers; legal specialists, including export control people for crypto; the PR department; and selected groups of technical people – for example, the original reporter of the vulnerability.

Decide when to publish the advisory. Do we have to wait for other vendors? What day of the week is it in other parts of the world? (Of course, if there is active exploitation, release as soon as possible.)

The advisory should use a vendor-independent format (text, HTML, PDF) and should be cryptographically signed. It makes sense to release an advisory internally first so that, for example, tech support knows what the customers are calling about! However, take into account the possibility of leaks, so pre-release only on a need-to-know basis, and/or don't allow much lead time; people need to prepare themselves, but too much time will increase the likelihood of leaks.

Once released, the advisory (at least the version on the Web) needs to be kept up to date; there may be corrections and additional information. When making changes, it is a good idea to update the revision number and keep a revision history so that people can decide whether they need to re-read the document.

KEYNOTE

A GLOBAL CULTURE OF SECURITY

Marcus H. Sachs, US Department of Homeland Security, USA

A "culture of security" is developing around the world which involves everyone, IT people and end users alike, and which is parallel to the "safety mind-set" that makes us wear seat belts in cars. The use of computers and networks puts any country at risk; that vulnerability must first be recognized before it can be addressed. In the early 1980s, AT&T ceased to have a telecommunications monopoly. Since then, the US telecommunications network is no longer domestic, terrestrial, and circuit-switched; there is a diversity of circuit- and packet-switched technology, terrestrial, satellite, and wireless, supporting voice, data, and other communications.

In the late 1990s, a military study determined that the Department of Defense could be reached from the Internet, and could be attacked through that route. Somehow, that information was at the time considered to be of only theoretical interest. A few months later, DOD computers became the subject of Net-based attacks. That got the attention of the leadership, and a task force was created to coordinate the defense of digital networks. At the end of that decade, there was a lot of cybercentric effort because of the impending Y2K, though for a while physical sectors of the infrastructure were somewhat neglected.

September 11, 2001, made it clear that the "physical" sector ought not to be ignored. When the two towers of the World Trade Center and some adjacent buildings were destroyed, the telecommunications redundancy that had been provided for the New York Stock Exchange turned out to be inadequate. Although connectivity had been purchased from several different companies, using several physical routes and going to two separate central offices, purchases and mergers among those companies had had the result that most of the "redundant" connectivity went through the same fiber bundles to the same C.O. In most large cities, telephones are connected to a single C.O., with little redundancy. On September 11, phone service in New York basically failed. Interestingly, people were still able to use the Internet to communicate. It was also found that the co-location of the various different utilities (water, gas, steam, electrical power) constituted a vulnerability.

The US government decided that "security" had been divided into too many little offices; the Department of Homeland Security was created to bring these functions under one umbrella. This department published two major documents in February 2003: "The National Strategy to Secure Cyberspace" and "The National Strategy for the Physical Protection of Critical Infrastructures and Key Assets," which are available from http://www.whitehouse.gov/homeland/. The Cyberspace Strategy is intended to be modular and to change as needed.

Why is it important to secure cyberspace? The USA is fully dependent on cyberspace, and the range of threats is huge, from script kiddies to nation-states. Recommendations include addressing vulnerabilities, as opposed to threats, because threats are constant and everywhere. Also, the government alone cannot secure cyberspace – individuals and industry must participate. The speaker went on to list and describe many initiatives being taken by the US government, and he listed requirements for a secure Internet: accountable addressing (such as IPv6); dependable network services (routing, DNS); trustworthy software; authenticated user services (Web, email); a working public key infrastructure; networks built to be secure from the start, not as an afterthought; the adoption of "best practices"; protection of and from "clueless users" making mistakes; the certification of network engineers; a mechanism for information sharing about computer security; and agreements between nations with respect to cybercrime.

TECHNICAL SESSIONS

WORST FEARS/WORKINGS OF A WORM

Roelof Temmingh, Sense Post, South Africa

There are over 1 million networked computers on Earth. The Internet covers not just the Western world, though the vast majority of connected computers are in the Western world, and Internet-based services include not only email and the Web, though those are the major services. Most of the computers are running Microsoft products. About 90% of Web servers run either Apache or IIS, and 96% of browsers are MSIE. The state of security ofn those products is not reassuring. Many vulnerabilities have been revealed in the past few years and have been exploited by some of the major worms. In fact, while the recent worms have exploited previously known vulnerabilities, most had no direct malicious payload or had an effect limited to DoS, and in most cases only a small percentage of targeted hosts were infected; nevertheless, their effect was large.

What, then, is the worst-case scenario? Imagine a group of 15 or so programmers working for six months in isolation.

Phase 0: Perform reconnaissance work (scanning) to find vulnerable Web servers (assume that we have prepared an exploit for an as-yet-unpublished vulnerability that will break Apache and IIS). Pick the 1000 busiest servers.

Phase 1: First we stealthily infect a thousand machines that are important, high-traffic Web servers (e.g., CNN). These "master servers" have huge log files and are constantly under attack anyway, so our attack has a better chance of remaining undetected, which is very important at this point. The code we inject into our victim "master servers" does DNS lookups of random names using DNS servers on predetermined controller hosts, and the return values of these DNS lookups contain commands for the master servers to execute. We send each master 1/1000 of the list of IP addresses of all vulnerable Web servers as determined during phase 0.

Phase 2: We now infect as many machines as we can, still stealthily. We send a signal to each master to inject attack code into random Web pages (but not the home page) of the Web server it has infected. Therefore, when a browser downloads that page, it also downloads the client version of the attack code. Using these "client servers," we do reconnaissance on the networks they are on (possibly internal networks not directly connected to the Internet), remove any antiviral software, collect any passwords we can find, and take control of any infrastructural machines (such as routers) that we can.

Phase 3: In a "pre-meltdown" phase, at a predetermined time (because communication with infected clients may be impossible), all "clients" look for vulnerable Web servers on the inside of that network and infect them so that they too will now carry the "browser worm," at least for the next three hours. At the same time, all "masters" carry out the infection of their predetermined lists of externally connected vulnerable Web servers.

Three hours after the above (which is short enough that it is unlikely that human intervention can take place in time to stop the process), the "clients" start destroying the local machine and network; they send DoS packets to any hosts that could not be infected, insert random bytes into data and text files of all kinds, corrupt or destroy the BIOS, and pop up a box asking users to call their local help desks, thus swamping the help resources and perhaps even the phone lines. At the same time, the "masters" remove all Web content, and start DDoSing Microsoft and Apache (to inhibit their supplying any patches), the root DNS servers, and random other sites.

Long-term data destruction (e.g., of data on backups) was also discussed, but it is a bit more involved.

The results of such a worm would effectively be total chaos. To prevent such a worm from succeeding, several measures should be taken:

- DMZ: Isolate outside-facing Web servers; don't allow machines in the DMZ to make connections to the inside.
- Tight filtering: A Web server should never initiate connections. If it is unable to initiate Web connections, it cannot spread a worm that way, even if it is itself infected.
- Internal segmentation: In case part of the organization gets infected anyway, the other parts are protected if the internal network is segmented by packet filters.
- Filter any third parties that have connectivity to your network.
- Use personal firewalls on user PCs as a last line of defense.

Automatic Exchange of Incident-Related Data and Its Application in CSIRT Operations

Klaus-Peter Kossakowski, Presecure Consulting GmbH, Germany

eCSIRT.net has a project whose goals are to improve the exchange of incident-related data and to foster improved cooperation among CSIRTs. For this, all groups involved have to agree on the meanings of words so that statistics are meaningful and a shared knowledge base is possible. The project also includes the provision of actual services related to this information, including "out-of-Internet" alerting so that information can get through even when the Internet is non-functional.

A code of conduct binds the participants in the information exchange, and covers issues of cooperation, protection of intellectual property rights and confidential data, and contributions to the goals of the project.

The common language is based on IETF initiatives Intrusion Detection Message Exchange Format, or IDMEF (to send attack information directly from the sensors), and Incident Object Description Exchange Format, or IODEF (for local CSIRTs to send a more high-level view of an incident). A Web form will allow constituents who don't yet support IODEF to send IODEF objects. IODEF will also be used to send information to local CSIRTs. IDMEF incident data can be included within IODEF incident descriptions.

eCSIRT.net has a clearinghouse function, to share as much information as legally can be shared; this is a low-priority task which occurs after incidents are closed, at which point they can be processed for statistical purposes. The output will be tailored to the recipient, where participating CSIRTs will get more detailed information, and the general public will get just an overview.

Three types of statistical information will be collected:

- Type 1 data are related to the workload of each CSIRT: number of attacks reported; false positives; systems attacked and affected; time spent analyzing, responding, documenting; and so on.
- Type 2 data concern incidents themselves. This information will be aggregated, and any identifying

information removed before it is presented to anyone.
- Type 3 data pertain to Internet events (which are not necessarily intrusions). These events will be monitored using automated techniques such as Argus (traffic monitor), honeypots, and so on. The event information will be accessible to constituents via HTTPS and user certificates.

The incident (type 2) information is highly controversial, because of trust and confidentiality issues; these issues are being addressed. There is potential that type 3 (event) information could have online processing resulting in alerts that can be sent to constituents.

There is technology to send out encrypted mail, to make phone calls and faxes, and so on. The idea is to free humans to analyze problems instead of tying them up in the mechanics of the transferring and storing of information.

Request Tracker for Incident Response

John Green, JANET-CERT, UK; Jesse Vincent, Best Practical Solutions, USA

RTIR is a tool for incident handling, which claims to be usable, cross-platform, open source, extensible, securable, and supported.

RT (the base software for RTIR) was designed to track issues of any kind. It gets used for bug tracking, help desk and customer service, network operation, "to do" lists, and so on. In its bare form, it does get used for incident response, but it is not ideal for this use. It is Web-based, and the client side is designed to work with just about any browser.

RTIR is an extension to RT, which uses the designed extension mechanisms; so, for example, it is possible to upgrade RT without having to "repatch" the extension. It is possible to add functionality, change the user interface, and so on. It should be run using HTTPS to improve security.

RTIR adds these functions to RT:

- An "incident" object ties together the various reports that might come in about a single incident and various actions such as blocking networks and performing investigations.
- Incident response team–specific workflows, for example, automatically opening incidents to notify the people responsible for each of a list of IP addresses about a vulnerability discovered while scanning.
- "Clicky" metadata extraction and tracking (to get more information about IP addresses through such utilities as whois, traceroute, etc.).
- Integration of whois information.
- Separate email threads for separate conversations about different tasks or components of the event.
- High-level overviews.
- Better searching tailored to incidents so that multiple events can be correlated.
- Simple scriptable actions.
- New reporting functions.

More information can be obtained from *http://www.bestpractical.com/*.

Communication in Software Vulnerability Process

Tiina Havana, Juha Roning, University of Oulu, Finland

Reporting software vulnerabilities is central to software development, but the communication process is problematic. In 2002, a study was done where vulnerability reporters and the receivers of such reports were questioned. Recipients reported contacting reporters more than the reporters believed that they were contacted.

The values and beliefs of the two parties differ. While they agree on the importance of security, precision and accuracy, and non-malfeasance (avoiding harm to others), reporters value public benefit and the public's right to know more than do the receivers, while the receivers place a higher premium than do the reporters on the avoidance of "FUD" (fear, uncertainty, and doubt).

Only just over half of the receivers passed on the bug information to their developers to prevent further occurrences of that type of bug.

One-third of the receiving organizations have a proactive publicity strategy for cases where there is a publicity crisis concerning vulnerabilities in their products, and one-third of them have PR personnel who are familiar with vulnerability issues and who have direct media contacts.

The vulnerability-reporting communication seems too often to be one-way; two-way symmetrical communication is needed. A dialog between the parties would improve mutual understanding, and vulnerability reporting policies would also be helpful.

PANEL DISCUSSION

Ask the Experts

Cory Cohen, CERT/CC, Carnegie Mellon University, USA; Robert Hensing, Microsoft, USA; Michael Warfield, ISS, USA; moderator: Roger Safian, Northwestern University, USA

Q: Concerning Microsoft's recent acquisition of an anti-virus product vendor, is Microsoft planning to automate anti-virus download-and-security-patch management into the same agent?

Rob: Microsoft is working on an anti-virus product of its own. It is also working with VIA (the Virus Information Alliance), and working on other security-related services geared toward home users. I cannot comment on the details.

Q: In the case of an organization with no real security other than that provided by volunteers, what "glory words" can be presented to financial folks to persuade them to devote resources to a security team?

Rob: To justify an IRT, use threat modeling: What am I trying to protect? What is it worth? A model called STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Damage Potential, and Exploitability) assigns dollar values to each of the named items. Chapter 4 in Writing Secure Code walks through the STRIDE model.

Andrew Cormack: Universities don't work on dollars and cents – list assets, liability.

Q: For home users and the general public, a major problem is spam. Break-ins on DSL-connected hosts are often motivated by the desire to install spam distribution agents. Various government organizations are interested in stopping this because it costs a lot of money. As security people, should we do something radical to the email infrastructure to address this issue?

Mike: Honeynet analysis of scanning for open proxies (squid, socks) shows that these scans are done mostly by spammers wanting to inject their spam. These guys have crossed the line at that point into illegal activities, therefore prosecutions need to be done. Spammers are trying to sell something, so they must leave ways to track them back. Our responsibility as security people consists of hardening email systems, implementing authentication between servers, and locking down open relays.

Cory: The growth of really usable cryptography could help. Membership in mailing lists should require use of cryptography to participate in them.

Roger Safian: The issues of someone abusing your resources versus someone just sending spam should be kept separate.

Rob: Technical approaches to this problem are seen here at conferences. Microsoft is chasing spammers legally' We need high-profile prosecutions to discourage this type of activity. Microsoft is suing 13 spammers, of which 11 are in the USA and 2 are in the UK. 60% of mail coming to Hotmail is spam!

Q: Lots of spam is fraud (eBay fraud, trying to get credit card information).

Mike: eBay and Pay-Pal frauds get much press coverage, but those make up less than 1/10 of a percent of the incoming spam, while one-third of it is for Viagra and other "personal enhancement" (!) technologies. Prosecutions need to proceed on the fraud front; this type of activity must become unprofitable.

Q: Regarding security advisories, is there any chance that advisory formats will converge so that automated means can be used to disseminate them, or their existence, to our constituents?

Mike: Advisory formats are evolving, but no wholesale conversion to a single common format should be expected. We may see two or three different versions of the same information (text, Web, XML), of which XML is most amenable to automated processing. But advisories must go through marketing and PR departments, which are less amenable to technical standards!

Cory: We are open to exploring standardized formats, but this is less important for advisories. We are more interested in a standard format for the exchange of vulnerability information, which can then be customized with text for advisories.

Rob: Microsoft released two new bulletins just today. After the advisories have gone through the legal people, they are mangled! If a standards body were to come up with a standard format for advisories, I would encourage Microsoft to probably play along. But Microsoft bulletins are already huge; a list of minimal information would be helpful.

Q: We have seen suspicious packets with a fixed window size. These were discounted by CERT/CC, while ISS went further and gave meaningful data. Give us some idea of your internal processes.

Mike: ISS got caught with its pants down on this one. We had noticed peculiar traffic, deferred looking into it for a week, then looked into it after incident reports had appeared in various other sources. At that point, we started detailed investigations, which are still in progress [as of 06/25/03], but postings have been made.

Rob: We [Microsoft] saw the traffic. No one could figure out its origin, and this caused great concern: Was it the first big kernel-mode Windows worm? [It turns out that this is probably not the case.]

Mike: This is a classic example of the fact that our business gets interesting with the words "I never saw it do that before!"

Q: What are the pros and cons of protocol- and signature-based IDSes? What are some of your favorite IDSes?

Cory: As a member of CERT/CC, I have no favorites, but we did some research on IDSes to assess their ability to really describe vulnerabilities using signature languages. We were disappointed about what could be done to really describe vulnerabilities. A lot of signature-based systems could not describe a recent Kerberos vulnerability, for example. Therefore, I have doubts about signature-based systems.

Mike: Marketing people want us to believe that the two kinds are very different. Protocol-based engines need more horsepower. Therefore it makes sense to try for the cheaper signature-based methods first to skim off 80% of the problem; then do protocol analysis to catch more; then, finally, there's the holy grail of "anomaly detection," which does not really exist yet. Each technique has its own domain of applicability; all of them should be used.

Rob: I'm more interested in host-based than network-based IDSes; the OS needs to get more intelligent. Tripwire, for example.

Q: Are there signature development classes, tips on developing signatures?

Mike: Get in good with the Snort guys, who are doing it all the time on our mailing list.

Cory: I found it difficult to match up individual signatures from various systems.

Roger Safian: We just completed testing eight IDSes of multiple kinds. All had plusses and minuses. Sales people are overzealous in their claims.

Q: What about intrusion prevention technology?

Mike: I have similar comments as for "protocol versus signature": there is a big area of overlap between technologies, and fuzzy definitions of them. Many vendors talk about putting things in midstream and blocking based on findings. There are concerns about false positives; we need a bit more track record on these devices.

Rob: Intrusion prevention is another way to say "host hardening": You can't attack what isn't there! I've been told that Windows listens on too many ports, that we cannot turn the services off. But Win2K has IPSec, which can be used to block access to ports.

Roger Safian: SANS cornered Gartner. It turns out that intrusion detection devices that are actually deployed are generally not used to block traffic!

KEYNOTE

The European Initiatives in Network and Information Security

Andrea Servida, Information Society Directorate of the European Commission, Belgium

Why Europe needs to act on security: 75% of European companies had no security strategy in 2002. It seems that underestimating core business risks is responsible for the low level of IT security investment in most European companies; there is a lack of awareness of the issues. The paradigm for security is changing from "security through obscurity" to "security in openness," but this openness and sharing of resources is a challenge to manage.

Toward a European-integrated approach to security: International cooperation is needed and is occurring in the following areas:

- Economic, business, and social aspects of security in an information society: These include business opportunities and growth, individual issues such as privacy and the protection of minors, dealing with the digital divide, and long-term preservation of knowledge and culture.
- Cybercrime, "homeland" security.
- External security and defense.
- Security research.

The role of the Commission: The European Commission proposes and orchestrates the development of a regulatory framework, for example governing electronic signatures, data protection in electronic communications (consent to collection of data, use of cookies), information, and network security. The Commission also launches policy initiatives, in particular to promote the development of various technologies or approaches.

eEurope 2005 and ENISA (European Network and Information Security Agency): eEurope 2005 build on the progress made by eEurope 2002: Internet penetration in houses has doubled since then, there is a telecommunications framework in place, and there is a very fast research backbone network. There are projects which affect CSIRTs, such as the European Information Security Program, which aims at providing SMEs with the IT security solutions needed to develop their e-commerce business. eE2005 aims to:

- Establish a cybersecurity task force by mid-2003. ENISA is not itself a CSIRT, but would have a facilitating role, coordinating existing capabilities and resources in member states.
- Develop a "culture of security" by the end of 2005 (develop good practice and standards).
- Secure communications between public services.

Ambient intelligence (wearable computers, computers inside our bodies) raises new security issues. Today's technologies are already pervasive and intrusive, and have huge interdependencies which are not being managed well. The challenge for tomorrow is to develop a respectful approach; the ethics of privacy will be a key element in an information society.

TECHNICAL SESSIONS

Introduction of the APCERT: New Forum for CSIRTs in Asia Pacific

Yurie Ito, JPCERT/CC, Japan

There is now a new forum for cooperation among CSIRTs in the Asia Pacific region: APCERT.

Most CSIRTs in the AP region do direct incident handling and coordination as well as issue warnings and alerts, technical bulletins, and so on. However, there is little participation from IRTs in industry. Many CSIRTs in the AP region do communicate directly with each other, to share observations, data, and technical information and to contact sites involved in an incident. While there are commonalities between the various CSIRTs, such as a narrow block of time zones and IP address blocks, there are differences in the technological maturity of the various participants.

There was a desire to coordinate the interaction between the AP CSIRTs. A working group was formed in 1997, whose core members were three large teams: CERTCC-KR, SingCERT, and JPCERT/CC. Over the years, it was decided to create APCERT, which was established in February 2003. Its objectives:

- Share security information among members.
- Handle security issues on a regional basis.
- Support the establishment of CSIRTs in countries not yet covered by such services.
- Collaborate with other regional frameworks, such as FIRST and TF-CERT.

There are currently 15 full members. APCERT's activities include an annual conference (APSIRC), and working groups on accreditation rules for APCERT membership, on training and communications for CSIRTs, and on financing the APCERT effort.

The APCERT involves other players, such as users, system integrators and operators, regulatory bodies, the insurance industry, law enforcement, and the technology development and engineering communities. It encourages these players to cooperate and communicate with each other; it facilitates mutual trust and information sharing.

Privacy Incidents on the Rise: Taxonomy and Response

Lance Hayden, Cisco Systems Inc., USA

What is privacy? As incident responders, we are called upon to take a "first responder" approach to privacy breaches, whether or not the direct causal link to computer security incidents is evident.

What is under attack? There is an evolution away from "computer security," where we are protecting information but we don't necessarily need to know what information it is we are protecting. We are now realizing that this information has a meaning, and can be used to cause serious damage to people and institutions. Attackers break into systems in general because they are looking for information; privacy is an extension of what CSIRTs have been doing as part of their jobs.

Identity theft is turning into one of the most prevalent crimes of this century. Criminals need anonymity (in the form of an identity other than their own), and stealing an existing identity that has a history is much more useful to them than creating a fake identity, which can be found to have "popped into existence" recently and can therefore be flagged as fake. This has implications beyond someone losing money from a stolen credit card, and we can expect civil and criminal liabilities for "enablers."

As the concept of "computer security" has evolved to "information security" and then to "privacy," the people responsible for working on securing that information have grown to include not only sysadmins, but also CIOs and, finally, individuals, who must safeguard their own information. There is a plethora of laws affecting privacy at the national and local levels.

The author suggested four basic categories (a taxonomy) of privacy breaches:

- Malicious attack: often preceded by security breach
- Process breakdown: security processes fail to protect
- Human or system error: not malicious, but still damaging
- Other

The following item apparently appeared on Slashdot the day of the presentation: The Palo Alto Unified School District had a wireless network which was not secured properly. A reporter for the Palo Alto Weekly parked in the parking lot and was able to pick up, for each student, full names and addresses, pictures, and in one case a psychological evaluation. While this could be characterized as a malicious attack, in fact errors in system configuration made the task of obtaining this information trivial.

The speaker described several more privacy breach cases, including one in which used computers were sold with their disks incompletely wiped, so that personal information about hospital patients (for example) was compromised.

Recommendations: Be prepared for the implications of privacy breaches. Apply not only deductive reasoning (what happened to cause this breach?), but also inductive reasoning (anticipating the impact of information loss).

Honeynets Applied to the CSIRT Scenario

Cristine Hoepers, Klaus Steding-Jessen, and Antonio Montes, NIC BR Security Office, Brazil

The Brazilian CSIRT set up a honeynet with the objectives of monitoring current attacks and intrusions, collecting data about opportunistic ("script-kiddie"-type) activity, developing new tools, and evaluating the usefulness of honeynets to CSIRTs. Requirements for the honeynet included low cost and reliability, as well as a high-quality data control mechanism. The team also wanted to make sure that it could prevent the use of the honeynet as a launching platform to attack other sites. Therefore, free software was used, and data was stored in "libpcap" format to facilitate its analysis with existing tools.

This team's honeynet started operations in late March 2002.

The honeynet's topology includes an administrative network, whose functions are to prevent outgoing attacks, to log activity, and to store artifacts and disk images of the honeypots. The honeypot segment includes several honeypots running different OSes.

Technologies used to control outgoing data (to prevent attacks from within the honeynet) include firewall rules (e.g., to block spoofed packets), outgoing traffic normalization (to discard invalid packets), a tool called "sessionlimit" (which can limit outgoing traffic based on fairly sophisticated state-based rules), bandwidth limitation to prevent DoS attacks from the honeynet, and an outgoing filter ("hogwash") to block traffic based on contents. Alerts and summaries are produced to report on activity in the honeynet.

Activities seen in the past year included lots of IRC traffic, a lot of worm activity (some new worms were captured), and denial-of-service attempts. Several new rootkits and exploit tools were captured. Statistics were produced on the top scanned ports (FTP, SSH, Telnet, and portmap were at the top). Also, many scans for open relays and open proxies were seen on ports 25, 1080, 3128, and 8080.

Worm activity concentrated on ports 80, 443, and 1433. There is still a lot of Nimda and Code Red activity, which means that many compromised hosts are still active!

The origin of the problematic traffic was graphed by country; for most cases, the US was at the top, though not in as large proportion as the US's presence on the Net. Back-door access and language of IRC conversations point to Romania as a major source of malicious activity.

By maintaining a honeynet, a CSIRT can provide an additional source of data to help understand what's going on in a particular set of incidents, and can pinpoint and notify compromised machines of constituents if they show up in the honeynet logs. This work is also a great source of training material for log and artifact analysis and forensic methods, and can be used to capture attack tools that would otherwise be only in the victim host's memory, since it is possible to capture full network traffic on the honeynet.

An Internet Attack Simulator Using the Extensions of SSFNet

Eul-Gyu Im, Jung-Taek Seo, and Cheol-Won Lee, National Security Research Institute, Republic of Korea

Because of the increase in Internet attacks, there is great demand for research on them and their effects. However, it is not always possible to study the attacks on a production network, for obvious reasons. This is why network simulators are useful.

SSFNet (Scalable Simulation Framework, network module) is a freely available network simulation tool which has a process-based discrete event-oriented kernel. The authors added extensions to SSFNet: a firewall and a packet manipulator. They then performed experiments with this setup; for example, they simulated a "smurf" attack in a network of 13,000 clients, 40 servers, and 270 routers, and were able to show the degradation of ping response times as a function of the number of subnets participating in the attack; it turns out that response times degrade drastically when 12 or more subnets are involved in the attack.

Policy-Based Configuration of Distributed IDS

Olaf Gellert, Presecure Consulting GmbH, Germany

IDS is a security component which monitors system events for unwanted behavior, using the following methods:

- Anomaly detection: As a first step, the IDS gathers statistics about normal behavior, and as a second step, it generates alerts on unusual behavior.
- Misuse detection: The IDS compares events against patterns of known attacks.

There are different kinds of sensors:

- Network sensors (NIDS) are components which inspect network traffic. One sensor per subnet is needed, and there is no feedback from hosts.
- Host sensors (HIDS) require one or more sensors (one per host), but they permit direct access to information.

An IDS also needs analyzing components to collect the data from the sensors, to take action on logged data, and to compile statistics. Management consoles are the front-end for visualization of logged data.

All IDSes suffer from false positives, false negatives, and often offer no explanation of an anomaly. With distributed IDSes, a large amount of generated data adds up: It's important to correlate all alerts for one attack into one single alert. With lots of sensors, the problem of configuring all those sensors becomes significant.

There is a diversity of different specification formalisms used for the configuration of IDSes, and different types describe different events. There are also configuration differences, even among several sensors of the same type, depending on their placement. In addition, anomaly-based sensors require frequent updates to their attack-recognition signatures.

There are several possible solutions to this configuration complexity, including some unacceptable solutions such as using only one vendor and/or only one type of sensor. There have been attempts to specify a single configuration language for all types of sensors, but this does not solve the problem of different configurations being required based on each sensor's placement within the network. The speaker suggested a solution: specify only a policy and generate rules automatically based on this policy.

The policy description suggested contains users, hosts, services, access by source/destination, and restrictions based on these combinations. In addition, one needs a database of rule sets to describe known attacks (for signature-based IDSes), a database of assets (existing systems, installed software), and a network topology database of some kind (not implemented yet).

A policy is used to generate a rule outline (a description of allowed services), based on which we use the asset database to generate rule specifications for each IDS based on their placement. Finally, we use the contents of the signature database (where applicable) to generate particular rules for each sensor. We can also set the severity for sets of events: for example, distinguishing between "alert for this event" and "just collect stats on this event."

The results of this work suggest that it is possible to configure all sensors centrally. Specific rules for each sensor reduce the number of false positives, and we see improved accuracy on the severity of alerts; it becomes easier to update the rules. A side benefit is that the assets database can be used for other purposes.

The maintenance of the asset database takes time, though this maintenance could be supported by automatic processes, using the output of the IDS, for example, or using scanning tools. Actual updates might still require manual confirmation, as might the signature database, though in that case the IDS vendors can help. Some unresolved problems: how to update rules without restarting sensors? how to introduce advanced topology information?

The status of this work: A tool to configure IPtables and Cisco access lists now exists (written in awk), and nearly completed work in object-oriented Perl is being done for handling subset-of-policy objects.

The Stealth File Integrity Checker

Frank Brokken, University of Groningen, the Netherlands

STEALTH (SSH-based Trust Enforcement Acquired through a Locally Trusted Host) is a file integrity scanner that has the advantage of being stealthy.

STEALTH's mission is to ensure the security of our computers (integrity, availability, confidentiality of information stored). Intruders may modify the integrity of the information on a host; our lines of defense include denying access when an intrusion attempt is detected, keeping software patches up-to-date, monitoring system logs, and, finally, knowing when relevant information is altered. This last is the goal of a file integrity checker.

File integrity checkers work by creating a "fingerprint" of the current state of a host, then detecting modifications of that fingerprint. The problem with the traditional approach of storing the fingerprint on the monitored computer itself is that the fingerprint itself is therefore vulnerable to intruders. The usual solution, to keep this state on read-only media, makes updating the fingerprint difficult or costly. The solution is to store the fingerprint on another computer, out of reach of the intruder, but in easy reach of the sysadmin.

The stealth "master" itself is not connected to the outside Internet, but can make SSH connections to its monitored clients. The fingerprint is stored by the monitor; there are no logs on the clients.

STEALTH itself uses standard software like "find" and "md5sum," so it is highly flexible and adaptable. The timing of STEALTH runs is unpredictable. Because the actual file signature calculations occur on the client, the monitor can be any old hardware; resource requirements are small. STEALTH can be obtained at *ftp://ftp.rug.nl/contrib/frank/software/linux/stealth/*.

KEYNOTE

Cyber Security in Canada

James Harlick, OCIPEP, Canada

Canada created OCIPEP (Office of Critical Infrastructure Protection and Emergency Preparedness) in February 2001 to enhance the safety and security of Canadians in their physical and cyber environments. The mandate has two components:

- Protection of critical infrastructure: physical and cyber components of the energy, utilities, communications, services, safety, and government sectors
- Emergency preparedness for all kinds of emergencies

Currently, the critical information structure is highly interdependent and has numerous vulnerabilities (e.g., a worm knocked out 911 service in part of the US because they were using VoIP). Canada has pledged itself to be the most connected nation on earth by 2006, which puts its infrastructure at great risk.

Canada's cyber-security framework contains four ways to reduce risks:

- Strengthen policy framework: Government departments and agencies are required to report cyber-threats and incidents to OCIPEP, and a framework is created to support information sharing and protection among various jurisdictions.
- Enhance readiness and response: protect (issue alerts and advisories), detect (coordinate identification and analysis), respond (establish incident response centers), and recover (provide incident impact analysis, technical assistance).
- Build capacities: This includes training and education (CSIRT training, training on malicious code analysis, IDS data analysis, assessment of vulnerabilities) and R&D (coordination with funding councils of the government, direct research projects).
- Build partnerships: It is necessary for partnerships to be formed internally, among the federal and provincial governments as well as critical infrastructure owners and operators, and externally, with other governments and CSIRTs. Already there is a daily "health check" information exchange between OCIPEP and the provinces, and there is a weekly conference call with critical infrastructure owners and operators in the private sector.

TECHNICAL SESSIONS

Multi-level Monitoring and Detection Systems (MMDS)

Madhavi Latha Kaniganti, D. Dasgupta, J. Gomez, F. Gonzalez et al., University of Memphis, USA

The presenter described an intrusion detection system (MMDS) which is an agent-based approach to monitoring and detecting attacks. The design is a hierarchy of specialized agents, with a fuzzy decision support system used to generate rules for attack detection. There are four types of agent:

- Manager agent: coordinates work and information flow.
- Monitor agent: gets information from sensors (a monitor agent should run on each host being monitored).
- Decision agent: uses information to decide what to do.
- Action agent: generates alerts and heartbeats. There is a GUI that can show graphs of various measured parameters.

The decision agent, which is the heart of the system, is based on fuzzy logic (fuzzy sets with "degrees of set membership" between 0 and 1). Instead of hand-coded fuzzy rules, rules are generated automatically by "training" the decision agent with genetic algorithms, under normal conditions and attack conditions.

Parameters monitored by MMDS include network activity (sent and received bytes and packets), user activity (logins, failed logins, number of users logged in), process information (total number of processes, number of root-owned processes, and number processes in various states), system parameters (physical and virtual memory in use), and MAC-level network data (sequence numbers).

Once the decision agent had been trained to recognize three attacks (SSH hack, nmap scan, and MAC spoofing on a wireless network), the researchers claimed very good results for detecting those attacks under test conditions.

Fire Your Firewall

Jan Meijer, Hans Trompert, SURFnet/CERT-NL, the Netherlands

The speaker's intention was to show that firewalls cause more problems than they solve (which is not to say that anyone has the right solution). The Internet (in 2002) was composed of 11,000 networks with 34,000 peerings, yet despite this complexity, it works! The use of firewalls tends to interfere with the proper functioning of the Net in many ways.

Firewalls break the Internet by misconfiguration. For example, Path MTU discovery, which permits the fragmentation of packets where necessary so they can reach a network with a "smaller" MTU, depends on particular ICMP control messages getting through – yet people do filter them (by filtering "all ICMP"). Other examples where a misconfigured firewall breaks things include jumbo frames, fragmentation, and certain applications, such as FTP, IRC, H323, IPSec, IPv6, and multicast.

Another reason firewalls break things is the added complexity they introduce: Because communication is no longer "end-to-end," it becomes difficult to find errors. Things no longer "just work" a lot of the time. The increased number of machines, people, and procedures involved means more opportunity for things to go wrong.

The time spent working around firewalls is time wasted (and sometimes workarounds are not available). For example, H323 videoconferencing requires four extra machines to work through a firewall!

Firewalls are single points of failure and will, of course, contain errors (since they are developed using the same processes used for other software!), and yet we place our trust in them.

Firewalls limit network speed by creating a bottleneck: Will firewalls, especially those which must traverse the protocol stack, keep up as networks speed up?

Firewalls consume scarce resources by requiring staff, equipment, time, and money to acquire and maintain. On a related matter, firewalls create bureaucracy: policy, carefully kept rules, lots of administration.

Firewalls create a false sense of security: There is usually traffic which works around or tunnels through the firewall (for example VPNs), and lots of traffic on legitimate protocols (email, Web) is still dangerous. Firewalls provide no denial-of-service protection. Too many configurations filter only inbound traffic; in this case, a "call-back tunnel" will get around the "problem," and local users do set these up. Even if there is a firewall, it is still necessary to patch, use end-to-end encrypted communications, switch off unneeded services, etc., measures which are too often neglected when there's a firewall.

INCIDENT RESPONSE AND THE ROLE OF LAW ENFORCEMENT

Kimberly Kiefer, Computer Crime and Intellectual Property Section of the US Department of Justice

The CCIPS, a 30-attorney section within the Criminal Division of the US Department of Justice, prosecutes computer crime and criminal intellectual property (IP) cases, trains and counsels agents and prosecutors, develops policy on computer crime and IP, provides input on legislation, and represents the USA on international bodies which address computer crime and IP issues.

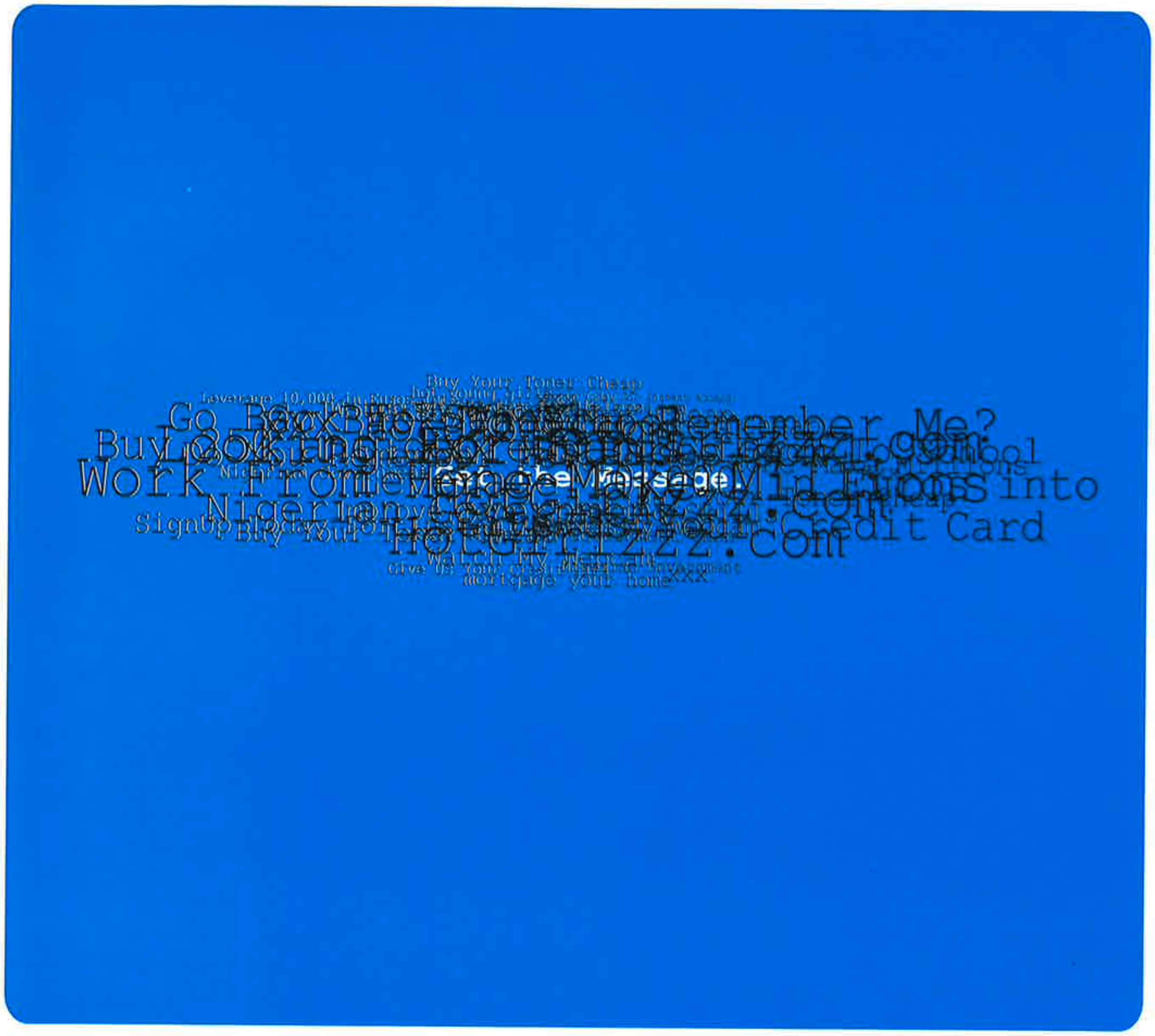
Security incidents are generally underdetected and underreported. Some reasons for not reporting incidents include the fear of negative publicity and competitive disadvantage, uncertainty as to whether law enforcement is interested or able to deal with the crime, not wanting to "challenge" crackers, and not knowing who to call or what is worth reporting. The speaker reassured us that law enforcement tries to be discrete with information and does not seize victims' computers; the victim does not lose control but, rather, is consulted closely. Also, the number of law enforcement agencies able to deal with computer crime issues has increased at all levels in the US.

There have been some success stories — for example, the successful prosecution of Mafiaboy (DDoS) and the Melissa virus author.

We are encouraged to report incidents even if the damages do not meet the police's criteria for investigation, because our incident may be linked to others, which collectively do meet the requirement.

Tips on cooperating with law enforcement:

- Keep detailed notes and logs, including records that will quantify the damages caused by the incident.
- Set up contacts with law enforcement before an incident occurs.



Get the Message.

Take control of your company's email gateway with PureMessage by ActiveState, the SMTP-filtering solution that stops up to 98% of spam, protects your network from viruses, and enforces your email policies.

See why ActiveState is a LinuxWorld Award double winner. Visit our booth at LISA or go to www.ActiveState.com to learn why PureMessage means real control of your email.

- Enterprise-scale SMTP filtering for AIX, FreeBSD, HP-UX, Linux, and Solaris
- Precise, logical, Sieve-based policy management
- Customizable anti-spam rules that leverage the open-ended power of a REGEXP-savvy engine
- Automatic or manual tuning of anti-spam technology
- Comprehensive logging, reporting, and quarantine interfaces

PUREMESSAGE by ActiveState

1.866.541.3182 **WWW.ACTIVESTATE.COM/PUREMESSAGE**

PRODUCT EXCELLENCE AWARDS LINUXWORLD CONFERENCE & EXPO

Active STATE

let us complain.

BEFORE YOUR USERS DO

Unbiased testing of HTTP intermediaries with hundreds of real-world and RFC-2616 test cases.

CoAdvisor

THE MEASUREMENT FACTORY

coad.measurement-factory.com

;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