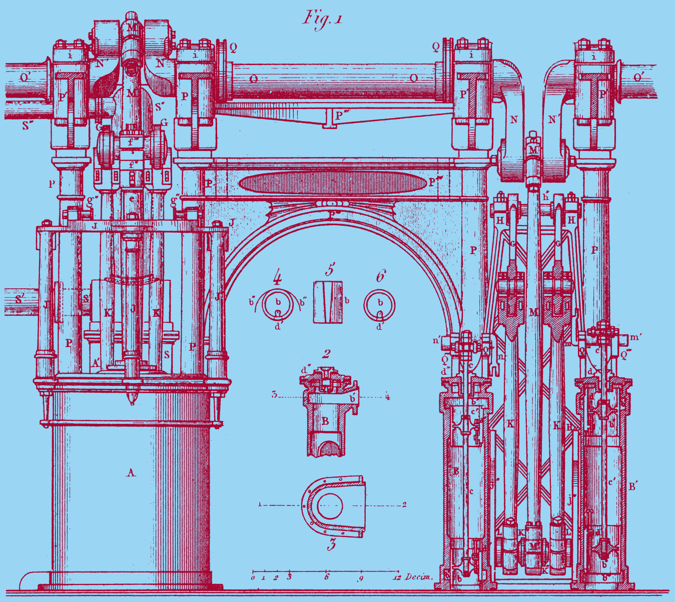
OCTOBER 2007 VOLUME 32 NUMBER 5

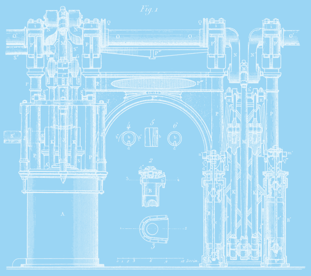
;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events



Internet Measurement Conference 2007 (IMC 2007)

Sponsored by ACM SIGCOMM in cooperation with USENIX

October 24–26, 2007, San Diego, CA, USA
http://www.imconf.net/imc-2007/

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

Sponsored by USENIX and SAGE

November 11–16, 2007, Dallas, TX, USA
http://www.usenix.org/lisa07

ACM/IFIP/USENIX 8th International Middleware Conference (Middleware 2007)

November 26–30, 2007, Newport Beach, CA, USA
http://middleware2007.ics.uci.edu/

2008 Linux Storage & Filesystem Workshop (LSF '08)

Co-located with FAST '08

February 25–26, 2008, San Jose, CA, USA

6th USENIX Conference on File and Storage Technologies (FAST '08)

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

February 26–29, 2008, San Jose, CA, USA
http://www.usenix.org/fast08

2008 ACM International Conference on Virtual Execution Environments (VEE '08)

Sponsored by ACM SIGPLAN in cooperation with USENIX

March 5–7, 2008, Seattle, WA, USA
http://vee08.cs.tcd.ie

Usable Security 2008 (USEC '08)

Co-located with NSDI '08

April 14, 2008, San Francisco, CA, USA

1st USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '08)

Co-located with NSDI '08

April 15, 2008, San Francisco, CA, USA

5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

April 16–18, 2008, San Francisco, CA, USA
http://www.usenix.org/nsdi08
Paper titles and abstracts due: October 2, 2007

2008 USENIX Annual Technical Conference

June 22–27, 2008, Boston, MA, USA
http://www.usenix.org/usenix08
Paper submissions due: January 7, 2008

2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)

Co-located with USENIX Security '08

July 28–29, 2008, San Jose, CA, USA

3rd USENIX Workshop on Hot Topics in Security (HotSec '08)

Co-located with USENIX Security '08

July 29, 2008, San Jose, CA, USA

17th USENIX Security Symposium (USENIX Security '08)

July 28–August 1, 2008, San Jose, CA, USA

8th USENIX Symposium on Operating Systems Design and Implementation (OSDI '08)

December 8–10, 2008, San Diego, CA, USA

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

contents

;login:

VOL. 32, #5, OCTOBER 2007

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
David Couzens
proofshop@usenix.org

PRODUCTION
Casey Henderson
Jane-Ellen Long

TYPESETTER
Star Type
startype@comcast.net

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

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

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

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

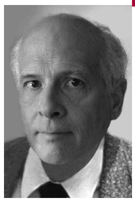
$85 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $120 per year.

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

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

RIK FARROW

musings



rik@usenix.org

Information technology, IT—the branch of engineering that deals with the use of computers and telecommunications to retrieve and store and transmit information.

—wordnet.princeton.edu

WHEN I FIRST READ ONE OF THIS issue's articles, I found myself suggesting to the author that he not use the term IT. I wrote back to him, saying that many of the readers of *;login:* are system administrators and that they don't consider themselves part of IT.

Having said that, I wondered how true that is. I also pondered over my own aversion to being considered part of IT. The definition of IT appears benign (see sidebar). It even appears to match, in very broad terms, what it is that system, network, and security professionals do.

The Priesthood

My earliest vision of IT came with my first visit to a computer room, sometime about 1961. I had volunteered to work with some "computer" people (no "IT" people back then) and write a program in assembly language that would run on a mainframe. The mainframe itself, in Rockville, MD, was tremendously impressive, with its many large, humming cabinets, the washing machine–sized disks with the intriguingly labeled (in red) "Write Protect" push buttons, and the CPU itself. I liked the flashing lights that showed memory addresses and data values, and it was sort of cool when the IT guy got the mainframe to play a song through the console speaker (poorly). But I was just as impressed by the spectacular air conditioning system, as the air inside the computer room was marvelously filtered, cool, and dry.

In 1969, I spent a summer working as an intern for GE in Bethesda, MD. I was the "software librarian," and the only really cool part of my job is that I got total use of the mainframe, during lunchtime, about once a week. I learned to boot the mainframe, starting with binary cards, then a larger deck of Hollerith-encoded punch cards, then bootstrapped the terminal concentrator, and finally got the tape drive, which contained the master copy of the OS and utilities, going. Once the system was up, I installed patches, from other nine-track tapes or from paper tapes, then dumped the patches to new nine-track tapes for shipment to the other 18 mainframes like this one around the world.

I took every CS course I could at the University of Maryland, which didn't even have a CS degree in those days. Using punch cards, and submitting jobs that you would not see the results of for at least 4 hours, and sometimes 20, really helped with typing discipline, as a single typo was tremendously painful. You would hand your card deck over to the priesthood who managed the CS department's mainframe, and they would load your deck into the card reader and then write it to nine-track tapes. The tapes, containing card images, got loaded on the mainframe, and the jobs would run one at a time. I later learned that there was some time-sharing going on, but most students were forced to run batch jobs. (With some 25,000 students and a single computer, we were fortunate that most were not doing anything with the computer.)

None of these experiences should have left me with negative feelings toward being part of IT. I did not work with computers my first several years out of college, because most jobs went to people with experience, and the Vietnam War had provided large groups of people with real experience, seriously impacting the market for those, like myself, who got college degrees instead of traumatizing experiences.

When I did get back into computers, it was because the microprocessor had started appearing in small systems. During the intervening years, DEC had been producing minicomputers, smaller, cheaper, and much less complex systems than their mainframe relatives, but even these were scarce. Microcomputers promised to change all that, making real computers available to everyone. I could see this clearly in 1978, and I quickly became a part of the early PC boom in the Bay Area.

Perhaps my aversion to IT came later, when I was working as a sysadmin consultant. I would have experiences where I would learn about some problem an IT staff was having and offer to take a look at it. For example, one company needed to migrate its database from a homebrew version to Informix. Both ran on SunOS, but the dump format of the homebrew version was incompatible with the import format of Informix. I talked to the IT guy who was struggling with the issue, figured out how the two formats differed, and wrote a short shell script that managed the conversion. All this took perhaps an hour.

The IT guy was amazed, saying that he had expected that it would take weeks of work for the IT department to do what I had done in an hour. I was perplexed. It really wasn't that hard, I thought, so what gives? Then I learned that such occurrences were the norm, and that getting anything done through the IT department was a bureaucratic nightmare that could take weeks or months.

Perhaps you can tell me if your own perceptions of IT are at all like mine. Perhaps you like being part of IT. I really don't know. I would be happy to release old, misbegotten perceptions, as the world moves much faster now, and I imagine that IT projects done by foot-dragging office drudges are now things of the past.

Lineup

We start out this issue with an article about Plan 9. Although Plan 9 may have faded from people's memories, the poster a group of HPC scientists had during USENIX Annual Tech generated considerable interest, so I thought it might be time to revisit Plan 9. Andrey Mirtchovski and Latchesar Ionkov don't want us to forget the many useful, yet simple, ideas that are part of Plan 9. In the case of Linux, the Plan 9 communications protocol, p9, has actually been ported and is still supported today. Plan 9 has much to offer, especially in the world of HPC, but also to students who need to be exposed to alternatives to the Linux OS design.

Cat Okita writes about Identity 2.0. I had heard Cat speak at LISA'06 and asked her to update her presentation for this issue. Identity goes beyond sysadmin, as it is something that affects every citizen of First World countries, and whose impact continues to expand.

Lex Muentz wants you to get ready for the possibility of litigation, as Federal Rules of Civil Procedure are morphing. These rules govern the handling of material required to be handed over during civil suits in the U.S. Lex clearly explains where the legal profession believes the landscape is changing. His advice on how to cope with electronic evidence will not only save you time and money but may help your organization avoid stiff fines that could be levied on you because you weren't prepared.

Octave Orgeron presents his second article about LDoms, the new Solaris virtualization capability. Logical Domains run above a hypervisor on some newer Sun systems, which does limit the number of people who will immediately be using these systems. But the techniques used in LDoms will, I believe, start appearing on other systems, as we already have other hypervisors, from VMWare and Xen. Octave's article explains how Sun goes about installing support, including firmware and software, for the LDoms system.

Hemant Sengar discusses some unsung dangers in VoIP protocols. His concern centers on the protocols used for setting up calls, protocols that can easily be abused for DDoS attacks and for DoS attacks with amplification.

David Blank-Edelman shows how you can draw pretty pictures, well, uh, diagrams, using Perl, and he highlights some easy-to-use libraries. David Josephsen describes how you can pry out information about what goes on within Java Virtual Machines. Debugging performance issues within JVMs requires access within the JVM. David explains tools that you can use to do this, as well as how to export performance data to your monitoring systems.

We have book reviews, of course, and Robert Ferrell explaining to us why we need to get rid of DNS. Yes, you heard me, but I will let Robert explain. Nick Stoughton has written an excellent article about the progress toward a new C standard.

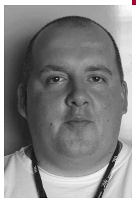
Dan Wallach tells the story of how the USENIX/ACCURATE Electronic Voting TechnologyWorkshop came into existence. If you have a new field of interest that you think might fit the workshop model, don't miss Dan's short article in this issue's "USENIX Notes."

We have two sets of summaries in this issue: the 2007 USENIX Annual Technical Conference and the Linux Symposium 2007.

It seems to me that my early experiences with IT did not create the antipathy I grew to feel later. But perhaps things have changed. What's your take on it?

ANDREY MIRTCHOVSKI AND LATCHESAR IONKOV

why some dead OSes still matter



Andrey Mirtchovski has been a Plan 9 aficionado since, as an undergraduate student, he found Plan 9's Third Release disks hidden in the waste bin of a university server room. Since then he has devoted a significant amount of time to the system, because it simply gets things done with less code.

andrey@lanl.gov

Latchesar Ionkov is a Linux kernel developer responsible in part for the 9p kernel module, which allows Linux to speak Plan 9's 9p protocol. Latchesar has been instrumental in translating Plan 9's ideas into mainline operating systems such as Linux.

lionkov@lanl.gov

WITH ITS CURRENTLY UNCHALLENGED ubiquity, the Linux operating system has become the de facto standard research OS in academia. The consequent waning of general systems research has been well documented [2], with the number of alternative operating systems in existence that significantly depart from the UNIX model and that are actively in development rapidly approaching zero. In this article we will discuss one such alternative operating system that has refused to disappear completely, and we will attempt to evaluate the features of that OS that make it suitable as a vehicle in our research projects and as a research environment.

We describe and share our experience working with the "Plan 9 from Bell Labs" operating system. This article will not attempt to compare the virtues of this OS with other, well-established systems. Instead, we will examine the environment Plan 9 provides as it pertains to researchers in academia, graduate students, and even undergraduates taking their first steps in exploring the inner workings of their first operating system. The goals of Plan 9 are completely different from those of other free and open source operating systems: Whereas some aim to provide a useful UNIX environment and others are bent on total world domination, Plan 9 aims to provide a useful research environment for building distributed software.

Plan 9 from Bell Labs

Plan 9 is a not-so-fresh offering from the same group at Bell Laboratories that created UNIX. Plan 9 has been in existence for over 18 years, making it a tad older than the popular free UNIX variants. The first Plan 9 papers were published in 1990 and since then there has been a relatively steady stream of research publications using the OS as their base [1]. Both the OS and its ideas are active parts in several current research projects, ranging from porting it to the largest supercomputer currently in existence to fitting it into small embedded devices that can be carried in one's pockets.

During most of its existence the OS has flourished within the confines of the lab it was created in, taking part as the core of most of the research projects therein. Outside of Bell Labs, however, Plan 9 has failed to gain widespread adoption. The reasons for this are several and can mostly be traced to the battle for open source software that raged throughout the 1990s. The first Plan 9 release was made available to a select few universities in 1990. A second release, made available in 1995, required external organizations and individuals to purchase the OS and manuals at the prohibitively high cost of $350 for a binary-only license. In 2000, Plan 9 marked its third release, this time as an open source operating system provided by its creators with all source code and at no cost. (Manuals can still be purchased separately from Vita Nuova [5], a company in Great Britain that maintains and distributes the Inferno OS, a cousin of Plan 9.) Unfortunately, the license Plan 9 was released under was slightly restrictive, for example requiring users to indemnify the new Bell Labs owner, Lucent, from any future lawsuits. As expected, this ruffled a few feathers in the free/open source camp (see Richard Stallman's article describing the issues with the license of the OS [3]). The license was completely opened and OSI-approved by 2002, for the fourth release of Plan 9. Since then the system has moved from a single release cycle to a continuous-update one, where changes to the OS and supporting applications are made available immediately, so the OS release number has not been bumped, even though the system is still being developed.

In many aspects the Plan 9 operating system was ahead of its time. Its creators anticipated the level of penetration that networks will have in computing and aimed to create a distributed environment that provided services to programs and end users regardless of their physical location as long as they were connected to the network.

PLAN 9 IS STILL RELEVANT

Even though Plan 9 has existed for nearly two decades, it still holds some relevance for today's operating system landscape. Plan 9 was innovative in many areas and integrated novel and interesting solutions deeply within the system, but its relevance does not necessarily stem from the fact that it's such a great OS: It is not. Other OSes that came into being about the same time, such as Amoeba [4], were making even greater leaps into the wild, containing a multitude of ideas, many of which—such as independent-of-origin computation, resource pooling, and virtualization—are now becoming much more relevant to computing. Perhaps that very boldness in accepting new concepts in the OS made those systems less practical; most of them have died from lack of developers and fresh users. Plan 9's fate is sure to be the same, as it has a relatively stable community of around several hundred users, but no new blood is coming in—which is exactly the rationale for this article.

Impracticality should be an accepted death sentence for every bold new operating system. What is troublesome, however, is that all the ideas coming from this vast research in distributed operating systems in the early 1990s have been lost to the new generation of programmers, students, and researchers. Students now entering the system software research and development field are faced with the same pool of choices now as they were 20 years ago: The choice is always among various UNIX offshoots. In fact, for many undergraduates the deciding factor as to which operating system they will dedicate their best years has been political, rather than technical: They pick from the set of free and open source operating systems the one that most closely matches their moral beliefs. The gap in operating systems and system software that was supposed to be filled by all that research in the last decade before the millennium has been filled by bloated middleware.

Plan 9 managed not to be involved in this political game. It came from an older time when source was closed and, when it had to, it successfully converted, with a few hiccups, to an open model allowing everybody to peruse its code as they see fit. What is keeping Plan 9 alive is those users who keep an open eye for problems and an open mind about their solutions. The reason Plan 9 is still relevant even after its user base has dwindled is that the main purpose for its existence is to explore and to examine problems and solutions in a networked, distributed environment. We can do so easily and without much effort because Plan 9 was built on three basic design principles that cohesively absorb and tie various parts of the system together: simplicity, clarity, and generality.

Simplicity

The initial goal with which Plan 9's creators set out to develop the system was to fix the problems that they perceived were inherent in UNIX. The main task they had to tackle was to simplify the system by peeling off all the communication layers that had been built on top of the core to handle tasks never envisioned by its creators, such as networking (sockets) or graphical user interfaces (X11). Plan 9 presents the programmer with a single protocol upon which all remote and local communication is based: 9p. The 9p protocol allows resources presented by processes locally or remotely to be accessed as a hierarchy of files and directories (which themselves are files) with a few standard operations such as open, close, read, write, and stat. The 9p protocol permeates the system fully, with absolutely all communication except memory access occurring over it.

Besides the fact that 9p allows clients and servers to share resources via a very simple but effective protocol, 9p has an extra feature that makes it very appealing to use: It is not transport-dependent. Plan 9 uses the protocol over TCP, IL, a protocol created by Bell Labs with 9p in mind and without TCP's overhead, and RUDP, a reliable UDP offshoot with in-order delivery. In our work here at LANL we have also written libraries that allow 9p to be transmitted directly over the DMA mechanisms available in the Cell for communication between its separate processor elements. We are currently working on a library that allows 9p to be spoken over the PCI-express bus in our next-generation hybrid supercomputer, which combines Cell accelerator cards with Opteron hosts.

We must mention that, apart from the simple communication protocol, the rest of the Plan 9 system is also an exercise in simplicity, in both the number of lines of code and the number of programs used to accomplish a task. The Plan 9 kernel for the PC architecture (by far the most used and most developed) consists of around 90,000 lines of C code, not counting the drivers for various hardware, plus another 20,000 lines of portable C code, which is shared among all architectures.

Given that this code is both readable and understandable, it comes as no surprise that new students are able to pick up Plan 9 rather quickly. Unfortunately, some students, particularly the ones already familiar with other free and open source operating systems, have expressed distaste for such a simple system. In our experience, undergraduate students tend to be more impressed by, even enamored with, graphical bells and whistles and are unable to give the system's simple design proper consideration. Indeed, most will consider a windowing system implemented in only 7000 lines of code spartan. What those students miss is that Plan 9's creators eschewed complexity, which left us with a system that is both easy to understand and easy to modify.

Clarity

The main method of interfacing with other programs is through files. A client program opens a file served by a process and reads to receive information or writes to send information. Plan 9 does not have an equivalent to the ioctl() system call, which cannot be issued across a network owing to the reliance on a local pointer to pass data. Instead, most servers by convention serve a file named ctl at the top of their hierarchy, which allows clients to control not only I/O but the general behavior of the servers.

The power of this method of exposed interfaces is enormous, as witnessed by the following examples, which illustrate how Plan 9's simple concepts are tied together into a cohesive environment.

The familiar concept of a UNIX pipe (i.e., a communication path linking a reader and a writer) is implemented in Plan 9 via a file server: A kernel device serves a single-level tree containing two files, data and data1. Writes to data can be read from data1 and vice versa. Moreover, those two files can be bound anywhere in one's namespace and even imported from a remote system, removing the need for a special tool such as netcat.

Networking in Plan 9 is also implemented as a file server, or, rather, several different file servers, each corresponding to a specific protocol. Those file servers are mounted together in /net on a system, and each may serve one or more subdirectories for each connection to a remote machine. Because Plan 9 directory structures can be served remotely and mounted on a remote machine (as the next section illustrates), a /net from one server can be used as the main interface of another. Since most networks nowadays are heterogeneous, this concept has mostly been used to provide access to the outside world to machines hidden on an internal network. One computer will have two interfaces—an internal one and an external one—with all machines on the inside that need access to the Internet mounting its external /net. Thus there is no need for specialized NAT software and packet translation. There are no extra provisions made to allow one machine to use another's network stack; instead, this ability simply comes from the fact that network stack interfaces are presented as files and that files can be imported from remote computers.

An extension of this idea is a small program called sshnet, which can be used to import a remote computer's network interface via a connection secured and encrypted by the SSH protocol. Sshnet imports the remote machine's TCP stack and presents it to the local system as a network stack like all others in /net.

The window managers and graphics subsystem in Plan 9 are also file servers. The window manager, rio, presents its control and communication interface in /dev. Those files correspond to the copy/paste buffer (a text file called snarf) and the whole screen of the display, as well as subdirectories containing information about each window on the system, including the text typed in and, indeed, the graphical representation of the window. To take a screenshot of one's desktop, one would simply have to `cat /dev/screen`. Since they're all simply files, naturally one can import another computer's window manager files and have local programs draw on a remote screen without any additional software requirements or tweaks. Furthermore, since they are all simply files, the window manager can be run recursively in one of its own windows.

Another example of the benefits of using files is the copy/paste buffer served by the Plan 9 window manager: It is a simple text file that acts as a buffer for text, keeping whatever is written to it for all readers. Naturally, to communicate with a more sophisticated operating system, this file is not enough. For example, when running Plan 9 under the Parallels virtual machine emulator on OS X, one is unable to have text copied on the host system be pasted in Plan 9. One ingenious solution to this problem is to export the host system's copy/paste interface as a file, which can be imported by Plan 9 and mounted on top of the file served by the window manager, thus being made available to all programs running within that namespace. The program to complete all this is a simple 100-liner in C which speaks the 9p protocol on a TCP/IP socket and knows how to query and set the copy/paste buffer in OS X. It serves a single root directory with a single file in it, replacing the Plan 9 window manager with the OS X one. This has proved much easier than having to create special hooks within both Plan 9 and Parallels to create an internal interface to the copy/paste buffer hidden from the eyes of user-level programs.

Generality

In Plan 9 every resource of a system, be it hardware or software, is presented as a hierarchy of files. The system provides two means of manipulating hierarchies: mounting a resource and binding two mounted resources together. These two functions are designed to improve and extend the ability of a Plan 9 user to construct a namespace (or the set of file hierarchies) relative only to the user's own interests and uses. In other words, a private view of what is available on the system (or networks of systems) is available to be customized by individual user-level processes on the fly.

Mounting carries the same functionality as it does on UNIX systems and their descendants: A connection with a local or remote resource is initiated, the resource is attached to the issuer's namespace (the directory hierarchy starting at root), and standard file operations pertaining to the mounted directory are sent over the connection to the server. In Plan 9, however, a mount can be accomplished by anyone without requiring superuser privileges and without affecting the file hierarchy of other users. This allows, as will be shown later, a user to import a remote server's network interface (à la VPN) in one terminal on their desktop without affecting any other users on the system or even programs running in other windows on the same desktop. Importing a remotely served file system is trivial: A connection is made to the remote system, and the file descriptor from that connection is mounted locally. The 9p protocol is designed to handle authentication security transparently to all programs accessing resources on the system.

Binding in Plan 9 is the ability to join two or more directories together in the same hierarchy. This does not have a direct analog in UNIX, but it has turned out to be very useful in Plan 9. It allows one, for example, to build a source tree in a directory that does not allow the user to write to it, without having to copy it to a temporary place. In another example, Plan 9 has only a single directory for binaries, called /bin. When a user logs in, all binaries for the particular architecture that he or she is using, all scripts (global and local for that user), and all binaries that the user has installed privately are bound to /bin. The shell's equivalent of $PATH in Plan 9 contains only a single directory, /bin (see Figure 1).

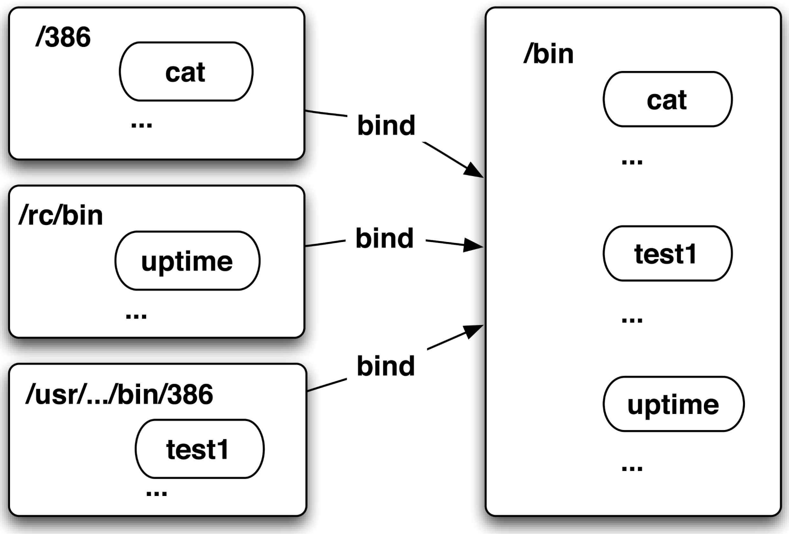


FIGURE 1: THE VARIOUS DIRECTORIES CORRESPONDING TO THE PARTICULAR ARCHITECTURE (IN THIS CASE, X86), SHELL SCRIPTS (RC), AND BINARIES COMPILED LOCALLY BY THE USER ARE BOUND TOGETHER IN A UNION DIRECTORY IN THE OTHERWISE EMPTY /BIN, THE ONLY DIRECTORY THAT THE SHELL TRAVERSES IN ITS PATH TO FIND BINARIES ON PLAN 9.

Perhaps the most striking example of the generality of Plan 9's ideas and how they tie together is the cpu command, which is used to connect to remote Plan 9 servers. We are used to thinking about such programs as gateways to remote machines. For example, when we ssh to a server we throw away everything on the local machine and accept the remote environment as our own. (The hack to tunnel X11 communications through SSH connections is the exception that proves the rule.) Plan 9's cpu command, however, makes it possible to connect not only two ports but two environments on two separate machines together. cpu serves the local namespace of the machine from which it was started under a directory (/mnt/term) on the remote machine. Since everything in plan 9 is a file, from there a script can automatically bind important files from the local system to where they should be expected by applications on the remote one. For example, the audio device is bound from /mnt/term/dev to /dev to allow any remote application to play audio to the local speakers. The same thing happens with the mouse and keyboard files and the graphical subsystem; thus any application run on the remote machine can draw to the local one. You can imagine this being extended for all devices: Via a simple one-line user command requiring no superuser privileges, one can, for example, print from any machine to a local printer.

As a consequence, Plan 9 system administrators do not set accounts and modify access lists for applications; instead, they modify file permissions, fully aware that if a user has permission to access a particular resource on a local system, that user can do so from any other computer that can connect to it.

EXAMPLES OF PLAN 9 IN THE REAL WORLD

The ideas stemming from Plan 9—namely, that all resources should be made available as files that can be accessed remotely by attaching them (or mounting, which is the more widely accepted term) to a user or to a process's namespace—have been put in use in various forms here at the Los Alamos National Laboratory's Advanced Computing Lab and elsewhere. Once one becomes acquainted with the system's simplicity and, dare I say, beauty, it is not very hard to convert ideas to fit the Plan 9 model of development. This has saved us a tremendous amount of time in developing and testing new protocols, implementing clients and servers, measuring performance, and scaling. The fact that we have based our tool on Plan 9's ideas, but have not directly used Plan 9 in some of the cases, is another benefit of Plan 9's design ideas and their flexibility: One need not port the entire OS but only one basic element, the protocol, to allow one to benefit fully from the research that went into the system. In all cases Plan 9's ideas have given us the ability to rethink from the ground up the basic ideas and principles on which our software was based, redesign it in a new model, and implement something that, if not much faster than its predecessor, was much simpler to understand and fix. Plan 9, in this case, served only as the "mind expansion" enzyme to further our work.

Some of our work, which is strongly based on Plan 9's ideas, includes:

- Xcpu: a job starter for extreme scale clusters. We created Xcpu at LANL in order to be able to start and control jobs on the next generation of high-performance computing platforms, which are starting to appear on the horizon. These machines will consists of tens of thousands of heterogeneous nodes. Xcpu presents a file server interface to starting jobs on a remote machine. This interface includes a directory for copying binary and data files; control files for starting, stopping, and continuing an application; and control files for attaching the application to its standard I/O and monitoring its progress remotely.
- CellFS: a new programming model for the Cell Broadband Engine which allows its accelerated components (also known as SPE units) to communicate with the host processor and its memory via POSIX-like file-based operations. Since DMA transfers to the host processor are encapsulated in 9p, the programmer simply issues an open request via a library call to access and a read request to fetch data from main memory.
- KvmFS: via 9p, provides extended access, startup, and control of virtual machines running on compute nodes across the cluster. Again, using a simple set of file operations, administrators can set up a virtual machine, transfer its image to a node on the network, start it up, and control its execution (including migrating it to a third node over the network).

There are numerous other toy programs and prototypes in which we have found the 9p protocol and the Plan 9 way of thinking, "everything is a file," to be of great help in simplifying and breaking down the problem space into its components. We believe now that the ideas we have learned from Plan 9 are applicable in wide areas of our research.

Conclusions

We are not trying to make the argument that Plan 9 should be considered by any and all academic researchers and students for their work. However, by listing here the ideas of Plan 9, we hope to invite students and operating systems programmers to keep an open eye for ideas and implementations. By showing what is possible with a little imagination and creativity, we described a system that will not be easy to conceive with ideas coming from the single-track mind of the successful, but antiquated, creations in the “real world.” We invite students and professors to look around and dig deep into the history of operating systems, especially the ones that never became commercial successes because they were “too far out there.” There is a great deal of research that went into new operating systems before the world got locked into commercialization, Plan 9 being only one such example, but without examining and evaluating those, we’re bound to continue making one mistake over and over: that of complexity. By building layer upon layer of interfaces designed to hide and sidestep the shortcomings of the underlying system, we are putting ourselves into the corner of incremental research, where our goal becomes that of improving what’s already there instead of throwing it away and replacing it with something better, guided by our experience and the ideas of others.

REFERENCES

[1] Plan 9 from Bell-Labs papers: http://plan9.bell-labs.com/sys/doc/.

[2] R. Pike, “System Software Research Is Irrelevant”: http://herpolhode.com/rob/utah2000.pdf.

[3] R. Stallman, “The Problems of the (Earlier) Plan 9 License”: http://www.gnu.org/philosophy/plan-nine.html.

[4] A.S. Tannenbaum et al., “Experiences with the Amoeba Distributed Operating System,” *Communications of the ACM*, vol. 33, pp. 46–63, Dec. 1990.

[5] Vita Nuova: http://www.vitanuova.com.

CAT OKITA

(digital) Identity 2.0

Cat Okita has more than 10 years of experience as a senior systems, security, and network professional in the financial, Internet, manufacturing and telecom sectors. Cat has spoken at LISA and Defcon about identity and reputation, co-chairs the "Managing Sysadmins" workshop at LISA, and programs for fun, in her spare time.

cat@reptiles.org

EVERYBODY "JUST KNOWS" WHAT identity is—most definitions center around identity as a "fact of being" or "defining characteristics"—but when push comes to shove, identity, like pornography, is very hard to describe exactly. Is your identity your name? Your government-issued photo ID? How you dress? What music you listen to?

What about the digital world? Is your identity your email address? Web site logins? How many "identities" do you have? In fact—is it *your* identity at all, if it's controlled and thrown around by other entities, with or (often) without your knowledge or permission? (Digital) Identity 2.0 is designed to change all of this confusion.

Identity 2.0 Is User-centric Identity Management

The basic idea behind Identity 2.0 is that it puts the users in control of their (digital) identity (or identities—think about the number of logins, memberships, and nicknames you have) and how their identity is used, managed, and given out.

"But wait!" you say. "What's this 'identity' thing anyway? Didn't you just say it's impossible to describe?"

Let's take a step back and talk about what identity is—and isn't. (To get a better idea of some of the issues, see Dick Hardt's informative and entertaining OSCON 2005 Keynote talk on Identity 2.0 [1].)

Defining (Digital) Identity

Since we're talking about computers and programs, I'm going to narrow our scope a bit and use the term "digital identity," rather than "identity," but digital identity as we're going to use it here is more than just an account name, IM handle, or email address. The following serves as a useful definition:

> A digital identity is a collection of claims attached to a digital subject (about which we can then make assertions).

A pithy summary might be:

Digital Subject A:
- Claims: TK421, short, stormtrooper.
- Assertion: TK421 is a stormtrooper.
- Verification: Stormtroopers must be tall—TK421 is short.
- Result: Aren't you a little short for a stormtrooper?

Common Problems with Identity

One of the wonderful things about digital information is that it's easy to copy, modify, move around, and destroy—and once a piece of digital information's been let out to the world, it's pretty much impossible to find out where it's gone and who's doing what with it.

Since it's so easy to copy digital information—and so hard to keep track of where digital information has gone—it's extremely important to know and limit who has access to what information. Of course it's much easier to say that we need to know and limit who has access to what information than it is to actually find out who has what information about you and determine how well it's protected.

In fact, the average G8 citizen is listed in 50–100 databases. Even if each database knows only a few things about you, combining information can reveal things you thought were private. It's far too easy to picture a world where Bob's insurance company decides to change what Bob's health plan costs just because they found out that Bob checked out a book about diabetes from the library and started to buy diabetic chocolate at the grocery store—even if he was actually trying to help Alice!

On top of that, who actually owns your information? If we think about medical lab results, do those results belong to you? Your doctor? The lab? Your (medical) insurance company? Some combination of all of the above? If that's the case, who gets to make decisions about sharing that information?

In fact, the usual way that you'd find out about a piece of digital information in the wrong hands is catastrophic failure—your credit card has been used to buy cell phone equipment on a different continent, or the government wants to have a word with you about tax evasion.

Myths About Identity

There are a few common myths that come up as soon as you start discussing identity—starting with the idea that there's a single definition for identity that everybody agrees on.

ONE TRUE NAME

> Everybody should have one (and only one) true name/nym. The name should be unique and identify the individual forever.

This myth usually stems from the naive idea that it's easy to come up with a scheme that will give each person One True Name—and that everybody will cooperate and play by the rules. It also relies on the availability of some sort of central system to track who owns what True Name and some sort of mechanism for tracking people down to prevent them from using a True Name belonging to somebody else.

There are a bunch of problems with this idea, starting with the need for universal adoption, how to enforce unique identifiers, maintaining privacy, and what to do if somebody steals or misuses your One True (forever) Name. And, in the end, should we really care whether a name is unique at all? Most of us manage to fumble our way through life without having terrible problems as a result of knowing two people named Dave Smith.

ONE TRUE ROOT/REGISTRY

The One True Root/Registry is often found as a close cousin of the One True Name, and it takes form around the idea that there should be some sort of global registry for identities, like IP addresses and DNS names. It's an interesting thought—but the sheer logistics involved in just registering 6.6 billion people are mind-boggling.

Some of the reasons suggested for the One True Root are:

- It ensures that identities are unique.
- It establishes a universally valid identity.
- It can be used for tracking and legal enforcement.
- It can reduce identity theft.

But who would be trusted to run this global database? What about privacy concerns and validating (or fixing) information? What should be done when conflicts arise (as they inevitably will)? These are all hard problems and completely aside from the intransigent requirement for universal adoption.

BIOMETRICS WILL SOLVE EVERYTHING

This is a nice way of saying "If we have your [DNA|iris scan|fingerprint], we can prove, beyond a doubt, that you're . . . uh . . . you." This is all very nice, but it's much like saying that you're never lost, because you always know where you are. Not only that, but there's no way to revoke biometric credentials, so once there's any sort of bad data (No-Fly lists, anyone?) associated with your biometrics, you're just plain out of luck.

So What Is User-centric Identity Management?

That would be one of the million dollar questions. Stepping back for a moment, let's take a look at the types of things that get called "identity management," and then move on to looking at the properties we'd expect to find in a user-centric identity management system.

There's a range of things that people mean when they talk about identity management. Jon Callas helpfully broke down the types of things that get called "identity management systems" into four main categories:

IM(1): Traditional ▪ AAA (authentication, authorization, accounting) ▪ Per-device user accounts ▪ PKI	IM(2): Traditional 2.0 (ways to make IM(1) easier, but all localized) ▪ Directory services ▪ LDAP ▪ Single sign-on ▪ NIS/NIS+ ▪ Kerberos
IM(3): Database Management ▪ Information management ▪ Metadirectories ▪ HR information resource management (phone numbers, titles, parking spaces, conference room reservations, etc.)	IM(4): Marketing ▪ Loyalty programs ▪ Buying habits ▪ Targeted selling ▪ Recommendation systems

All of these systems have a few things in common. They are local. They are controlled by a single authority, such as an employer or a retailer. Moreover, information about the user is controlled and managed by an authority other than the user—in many cases, users may not even know what information about them is in the system.

What Properties Should a User-centric Identity Management System Have?

If we look at Identity 1.0 management systems, it's pretty clear that they're far from user-centric. In fact, users have little to no control at all over their information, who has it, and what's being done with it. This obviously raises questions about privacy, security, accountability, trust, and manageability.

A number of smart people have written (at length) about the properties a user-centric identity management system should have. I'm firmly of the opinion that one of the key properties is usability. It's been proven time and time again that people reliably screw up complicated things, so if it's not easy for users to manage and understand what's being done with their information, it's like handing them a genie in a bottle. It's easy to let the genie out of the bottle but awfully hard to put the genie back (if you can at all!).

One of the most widely advertised lists of properties for a user-centric identity management system comes from Kim Cameron of Microsoft. Unfortunately, Kim's "Laws of Identity" [2] are a miserable failure when we're talking about something anybody can understand. Here are his seven laws:

1. User Control and Consent
2. Minimal Disclosure for a Constrained Use
3. Justifiable Parties
4. Directed Identity
5. Pluralism of Operators and Technologies
6. Human Integration
7. Consistent Experience Across Contexts

Dr. Ann Cavoukian [3], Ontario's Information and Privacy Commissioner, took Kim's laws and produced a notably clearer (although still rather long-winded) interpretation:

1. Personal Control and Consent
2. Minimal Disclosure for Limited Use: Data Minimalization
3. Justifiable Parties: "Need to Know" Access
4. Directed Identity: Protection and Accountability
5. Pluralism of Operators and Technologies: Minimizing Surveillance
6. The Human Face: Understanding Is Key
7. Consistent Experience Across Contexts: Enhanced User Empowerment and Control

Ben Laurie [4] of Google produced a nicely succinct set of properties for an identity management system that are easy to understand and remember and are a great place to start thinking about what we want from user-centric identity management:

1. Verifiable
2. Minimal
3. Unlinkable

I always like to make the implicit need for systems that people can use explicit and add a fourth property to Ben Laurie's three:

4. Usable

What Do These Properties Mean?

I'll cheat briefly and use Ben Laurie's pithy properties and their associated comments as a starting point.

1. Verifiable: There's often no point in making a statement unless the relying party has some way of checking whether it is true. Note that this isn't always a requirement—I don't have to prove my address is mine to Amazon, because it's up to me where my goods get delivered. But I may have to prove I'm over 18 to get the alcohol delivered.

In other words, can we prove enough about what you're claiming to believe that it's true enough and, if we can't, who's accountable for the bad information?

2. Minimal: This is the privacy-preserving bit—I want to tell the relying party the very least he or she needs to know. I shouldn't have to reveal my date of birth, just prove I'm over 18 somehow.

If we're going to take control of our identities, we have to start by not handing everything about ourselves over to anybody who asks. In security terms, this means deny all and permit selectively. In identity terms, this means anonymity with selective (and minimal) disclosure.

3. Unlinkable: If the relying party or parties, or other actors in the system, can, either on their own or in collusion, link together my various assertions, then I've blown the minimality requirement out of the water.

Of course it's impossible to be either anonymous or selective if it's easy to put together a few claims and figure out who you are or what you've been doing. I'm sure everybody's had the experience of their mom pointing out that they can't possibly have gotten mud all over if they stayed inside all day.

4. Usable: If I can't figure out how to use this system, or it's easy to do the wrong thing (whether that's giving out my information to everybody or letting somebody steal it), it doesn't matter how good the system is at everything else.

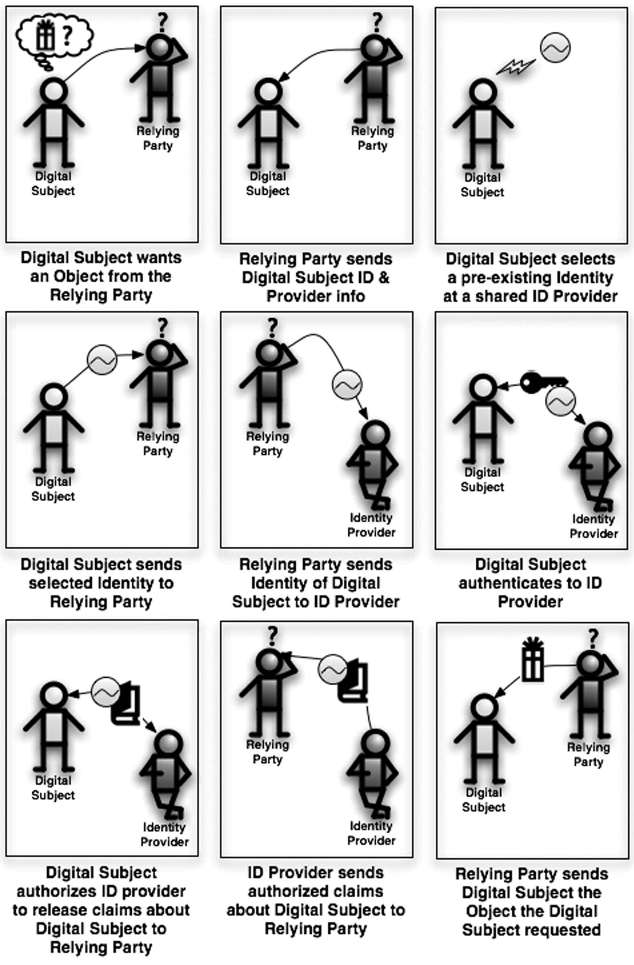
In the end, all of this is helpful only if it's easy to understand, and anybody can figure out how this whole user-centric identity management thing works—and do it without losing their identity.

All That Aside, What's Identity 2.0 Good for, Anyway?

You'd think that Identity 2.0 being user-centric identity management would mean that it's all about the user—and you'd be partly right. Identity 2.0 is definitely about the user, but there's more than just the user in the mix.

The basic architecture of identity management 2.0 has a user (Digital Subject) with one or more sets of characteristics (Identities), contacting a service provider (Relying Party), to obtain a service, which is authenticated and/or authorized by an identity management service (Identity Provider) on behalf of the user.

That means that we've got users, service providers, and identity providers, all of whom have their own vested interests. Eliot's dad wants to stop having to remember tons and tons of online user IDs and passwords; MIT wants to be able to share library logins with Harvard; Dan wants to be able to buy smut without having to give away his birthday; whitehouse.com wants Dan


Digital Subject
Relying Party
Digital Subject wants an Object from the Relying Party
Digital Subject
Relying Party
Relying Party sends Digital Subject ID & Provider info
Digital Subject
Digital Subject selects a pre-existing Identity at a shared ID Provider
Digital Subject
Relying Party
Digital Subject sends selected Identity to Relying Party
Relying Party
Identity Provider
Relying Party sends Identity of Digital Subject to ID Provider
Digital Subject
Identity Provider
Digital Subject authenticates to ID Provider
Digital Subject
Identity Provider
Digital Subject authorizes ID provider to release claims about Digital Subject to Relying Party
Relying Party
Identity Provider
ID Provider sends authorized claims about Digital Subject to Relying Party
Digital Subject
Relying Party
Relying Party sends Digital Subject the Object the Digital Subject requested

to be able to buy smut easily (and without worrying about his privacy or credit card numbers going missing); the government wants to be able to identify and serve constituents without violating their civil rights; and Verisign loves the idea of providing yet another registry service. It sounds like everybody's a winner here.

Of course, the big winners in all of this focus on Identity 2.0 are the middlemen. We've got plenty of users and service providers already. The most common design for implementing Identity 2.0 does a great job of creating new business for identity providers and middleware brokers.

So, Who's Doing This Stuff?

We've got all of the usual suspects involved—big corporations on their own or in groups, the free/open source community, standards bodies—and a variety of interesting implementations.

There's been a remarkable convergence in the Identity 2.0 space over the past year. Microsoft's CardSpace is still holding down one corner. The Liberty Alliance (composed of almost everybody other than Microsoft) has formed up in another. OpenID is being cheerfully adopted by the Web 2.0 crowd and is also collaborating with Microsoft. And everybody's using (or at least supporting) SAML (the OASIS Security Assertion Markup Language). Table 1 provides a summary of those involved.

Other nontraditional players, such as Google, Yahoo!, and Amazon, are also sneaking into the identity management space, with APIs that enable single sign-on for their properties or redirect authentication queries to third-party servers.

	Verifiable	Minimal	Unlinkable	Usable	Comments
Microsoft CardSpace	Yes	Yes	No	?	http://cardspace.netfx3.com/ Requires Web Services Trust Language Typically requires additional middleware Aimed toward end-user e-commerce
Liberty Alliance	Yes	?	?	?	http://www.projectliberty.org/ Aimed primarily at enterprise use cases Single sign-on/logout, permission-based attribute sharing, circles-of-trust, interoperability certification Uses SAML
OpenID	?	Yes	?	Yes	http://openid.net/ Low/no trust authentication only Distributed free Lightweight (by comparison, at least) Actively in use, primarily with blogs (e.g., Six-Apart, Wordpress, and a variety of blog software, including moinmoin, drupal, phpBB, and mediawiki)
Shibboleth	Yes	Yes	No	Yes	http://shibboleth.internet2.edu/ Uses SAML Web single sign-on only Most common in academia
Pubcookie	Yes	Yes	No	Yes	http://www.pubcookie.org/ Web single sign-on within an organizational domain Most common in academia
Google Apps					http://google-code-updates.blogspot.com/2007/02/new-apis-for-google-apps.html Allows authentication to be redirected to a third party for hosted apps http://code.google.com/apis/accounts/AuthForWebApps.html Allows applications to authenticate against and use Google services
Yahoo! BBAuth					http://developer.yahoo.com/auth/ Allows external Web applications to authenticate against and use Yahoo! services
SAML					http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

TABLE 1: PLAYERS AT A GLANCE

Are We There Yet?

Is Identity 2.0 mature—or even walking yet? Not really. We're starting to see OpenID implementations popping up all over, and there's enough convergence in the Identity 2.0 space to suggest that we're starting to hit critical mass.

Unfortunately, there are still plenty of outstanding questions about security and why you'd want to trust a third-party identity provider (or providers) with your information. It's certainly true that you'd have fewer places where you'd have to remember passwords (or some other form of authentication), but that also means that the identity providers become richer targets, and it certainly makes collusion among providers (or relying parties and providers) much, much more interesting.

Beyond that, is it really user-centric identity management when you're still trusting and relying on third parties to do the right thing? Or is it just rearranging to whom you've contracted the outsourcing of your identity?

REFERENCES

[1] Dick Hardt gives the best talk on Identity 2.0: http://www.identity20.com/media/OSCON2005/.

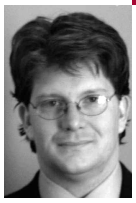
[2] Kim Cameron, "The Laws of Identity": http://msdn2.microsoft.com/en-us/library/ms996456.aspx.

[3] Ann Cavoukian, "7 Laws of Identity": http://www.ipc.on.ca/images/Resources/up-7laws_whitepaper.pdf.

[4] Ben Laurie, "Laws of Identity, Revised": http://www.links.org/?p=222.

ALEXANDER MUENTZ

hardening your systems against litigation



Alexander Muentz is both a sysadmin (since 1999) and a lawyer (admitted to the bar in Pennsylvania and New Jersey). He'd love to be a hacker public defender but has to earn his living helping law firms do electronic discovery. When he's not lawgeeking, he tries to spend time with his wife and his motorcycle.

lex@successfulseasons.com

YOU SPEND ENOUGH TIME WORKING in IT, and you'll go to a conference. You get to see distant friends and colleagues, get free stuff, and maybe learn a thing or two. You come back with a headful of ideas. Sometimes a FUD-wielding salesperson actually gets you worried and convinces you that you need to fix something back home. There are times when it's justified, and other times when it's silly.

The lawyers who represent your organization are going to their own conferences, and there's a big scary thing on their radar. To make your life easier, I'm going to explain what they're worried about and how to prepare your systems for litigation. For brevity's sake, I'm lumping technology support roles such as system administration, network engineering, and user support into "IT." I'm both an IT professional and a lawyer, but don't take this as legal advice.

What's the big scary thing? Recent amendments to the Federal Rules of Civil Procedure (FRCP) [1] have changed what digitally stored information must be retained and turned over to other parties to a lawsuit. The FRCP are the ground rules for civil litigation in federal courts. Generally the largest and most complex lawsuits are heard in the federal system, and state court systems tend to look to the Federal Rules when they modify their own rules.

Substantial changes to the FRCP are rare, which is why there's lots of buzz right now. Every electronic discovery vendor is out there teaching Continuing Legal Education classes (CLEs) about what the new rules are, what they mean, the big scary sanctions, and why you should hire its firm to handle all the digital aspects of your lawsuit. I've been to a few of these CLEs and taught one of my own, so I can safely claim that other than forcing lawyers to sit down and read the rules over lunch, they're really not that useful, because no one knows what the rules truly mean yet.

A Quick Explanation of Judicial Review

Often rules and statutes are intentionally left vague to keep pace with changing social, economic, or technological conditions without having to revisit the basic law. To fill in the gaps in interpretation, judges look to previous decisions on similar facts, which may look to even earlier decisions. After a few iterations of this process, whole new doctrines known as "judicial gloss" are created to answer any contingency. To fully understand the rule, often many judicial opinions must be read. This makes our law stable and predictable, but arcane to the nonlawyer.

There have been so few decisions on these rules that it's like trying to learn what's going to happen in a movie by watching the credits. Unfortunately for the lawyers, without judicial gloss to guide us, we can only guess at what they really mean, and that worries us.

And for good reason. Not only are the rules still open to wide interpretation, but getting it wrong can have serious consequences. For example, the FRCP now require that a party to a lawsuit make available any Electronically Stored Information (ESI) that is both "relevant to the claim or defense of any party" and is not privileged, cumulative, or "reasonably accessible due to undue burden or cost." Improperly withholding or destroying discoverable information can result in sanctions, either monetary or the exclusion of your side's favorable evidence. Until the women and men sitting on appeals courts start handing down opinions in a few years and giving us some more fleshed-out rules to follow, we will only make educated guesses to protect our clients. Adding to the anxiety is the generally low level of technological savvy among lawyers.

FRCP Affects Sysadmins and Other IT Professionals

The FRCP require parties in litigation, or to whom litigation is likely, to immediately preserve all information under their control that tends to support their claims or defenses in the lawsuit. Failing to preserve or intentionally destroying such information can result in sanctions. Once litigation is started, all parties must provide either a list of sources and locations of relevant information or the actual information to the other parties in a process known as *discovery*. Information, if delivered, must be provided either in the format originally used in the course of business or in a format agreeable to both sides. Parties can also request additional disclosures of relevant information and can attempt to force the other side to deliver information improperly held back. They can also require uninvolved parties to divulge relevant information under their control. Each side also has a duty to inform the other side if it locates additional relevant information.

Generally, there are few limitations on such disclosures. Information that is legally privileged, redundant, a trade secret, or classified can be withheld unless a judge orders its disclosure. Also, a party can claim that the information sought is "not reasonably accessible" due to effort or expense relative to the value of the lawsuit [2]. If the requesting side pushes the issue, it may get it but be forced to pay costs to make such information usable, such as for data recovery of damaged media, media or format conversion, or forensic analysis for deleted files.

Sanctions: The Scary Bits

What the lawyers are really worried about are discovery sanctions. Once litigation is foreseeable, intentionally or negligently destroying discoverable material can result in monetary or evidentiary sanctions. Monetary sanctions are normally limited to the legal bills expended in forcing the disclosure of discoverable information. Evidentiary sanctions can be uglier. Information improperly withheld may be barred from being used in court by the side that withheld it or may even result in loss of a claim or defense, which can lead to losing the entire lawsuit [3]. There is a "safer harbor" provision in the FRCP that stipulates that any destruction of discoverable information resulting from the normal, automatic operations of your systems is not sanctionable. Right now that seems to be only relevant to operations with little user control—overwriting deleted files, rolling logs, and the like. Human-controlled but routine procedures such as tape rotation may not be covered here.

The Definition of "ESI" Is Open to Interpretation

The definition of ESI is intentionally vague, to incorporate any new technology, but 99% of e-discovery revolves around email and "edocs." Email is what you think it is, and edocs are the entire spectrum of end-user documents—MS Office files, CAD/CAM files, images, PDFs, and the like. Given the way in which most discovery is handled at large law firms, I like to call this the "presumption of printability." If the file can be printed, they're most likely interested in it, because that's what they're doing, in one form or another. IT-savvy firms are converting the files to .tiff and using an image database to host them for armies of lawyers to look at each email or document. But if it's something not immediately understandable to a human, such as a database file, or not printable, such as a sound or movie file, the big firms tend to set them aside unless they have a good reason to look at them.

Few lawyers get creative with their electronic discovery requests. I understand not wanting to demand all of the stored music or movie files on an opposing party's PC (unless you're the RIAA or MPAA), but many lawyers are resistant to requesting archived voicemails, IM chat transcripts, and the like. Every CLE that I've seen or taught on this issue has touched on the multitude of potential storage devices, including employees' home computers, mp3 players, thumb drives, PDAs, and cell phones.

ESI may even be information that isn't normally stored at all. A recent federal magistrate's decision in *Colombia Pictures v. Justin Bunnell*, commonly known as the Torrentspy ruling, upheld a litigant's request to force a Web site owner to keep and preserve requesting IP addresses when he originally did not store that information [4]. The responding side failed to prove that the logging would be unduly burdensome, as it was using Microsoft's IIS, which can easily log such information. Lots of people in this field are eagerly awaiting future rulings to see whether this rule is followed or ignored. Stay tuned.

How Lawyers Process This Information

A typical large litigation project works in stages. First, the lawyers determine what information is likely relevant to the litigation and which employees would have this information. They then collect it from their client and have armies of lawyers pore over it, both to get an understanding of the facts of the case and to mark any documents that may be excluded because of legal privilege, trade secret protection, or irrelevance. Once that is done, they start handing over this information to the other parties to the lawsuit and start looking at the information given to them by the other parties.

Where You Fit Into All This

Some background is useful to understand what the lawyers really want and need. You're important because most of this information is on systems that you're responsible for. You or your group is most capable of collecting and preserving the data your organization needs. To assist, I'm going to give some suggestions of what you should be doing, depending on where your organization is in the litigation process.

STAGE 1: NORMAL OPERATION—NO FORESEEABLE LITIGATION

This is the time we like to think of as "project time." There are no major fires to put out, so you can think clearly about long-term fixes for issues such as litigation readiness. I'd recommend getting a handle on all the places where your organization stores data. Make an inventory of all your workstations and server shares, both active ones and the obsolete ones on the shelf. Figure out which users have or had regular access to what workstations, shares, databases, and other resources. If you've got some time and budget, look into archiving and indexing packages for email and end-user documents. If you support end users, look into remote control/login packages that allow you to collect locally stored data and email over a network.

Also, this is the time for you to work out your data retention and destruction policies. Work them out with everyone involved: the IT staff that implements and maintains it, as well as the legal and compliance crew that makes sure it's compatible with your legal and regulatory environment. Once you've got it all in place, stick to it. If something is supposed to be deleted, preserved, or destroyed, do so. Surprises during litigation aren't any fun.

For the desktop support crew, a decommissioning procedure, during which you index and archive the user files when a workstation or employee moves on, keeps you free from worrying about trying to find and resurrect obsolete machines a few years later. Doing the same for server shares that are no longer active also makes good sense. Having at least one working system that can read whatever media you chose to store those archives on will also make your lives easier, unless you like scouring eBay for antiques.

STAGE 2: REASONABLE LIKELIHOOD OF LITIGATION

A bunch of employees just jumped to your largest competitor, maybe with the products they were developing. Your organization is seriously considering suing someone. Ideally, your lawyers write up a nice, straightforward litigation hold memo and circulate it to the IT staff. Now your homework from Stage 1 pays off. With your list of shares, workstations, and users to preserve, all you need to do is keep the archives in a safe place, fire off a full backup of the current affected users, shares, and resources, and run incremental backups until the litigation hold is modified or ended. If you haven't prepared, you need to scramble. The legal department will need to send letters to all the affected users asking them to stop deleting relevant documents and email. You may get asked to enforce this, somehow.

STAGE 3: LITIGATION COMMENCES

Someone's filed suit. First off, the litigation hold may be modified and expanded. Second, you need to start putting all of your relevant ESI into two categories—easy and hard to produce. Your lawyers are going to want to know which ESI is expensive and time-consuming to hand over to the other side, in order to exclude it or force the other side to pay for the extraction. For each item you're preserving, come up with a rough estimate of the time and money it would cost. Be realistic: 4 GB of MS Office files in an end user's home directory isn't going to take ten hours and a thousand dollars to produce, but that crate of 9-track isn't going to be cheap to convert into some modern, usable format, either.

Finally, you have to prepare for the "Rule 26" conference. FRCP 26(f) requires parties to sit down and work out a discovery plan. Included in such a plan is how to handle ESI: what format to give the discoverable information in, when to supply it, and where to deliver it. Each side should bring an IT professional who understands what information is sought, what format(s) it is in, and how difficult it would be to make it available to the other side in the format requested. This IT pro should also be able to explain technical concepts in simple, easy-to-understand language without getting frustrated.

If you haven't prepared for this during Stage 1, get ready for long nights and stress. I've worked a case where the client had no capability for remote archiving, so I had to travel to four separate cities to personally collect documents and email. If there had been old workstations or tapes that could have held discoverable information, I'd probably still be pulling data from them today.

STAGE 4: HANDING OVER DISCOVERY TO THE LAWYERS

The parties have started to agree on the scope of discovery, so you can go back to your archives and start pulling out information that complies with this. Now any indexing or searching capability you have is going to come in handy. The lawyers may have agreed to hand over any documents or email that contains a project name, or falls within certain dates, or to specific people. The ability to search and to limit the amount of discovery is going to reduce your legal bill, as the fewer documents your own lawyers have to review, the less billable time. In addition to the delivery of documents, there may be some assistance you can offer to the legal team in sorting and understanding the discovery that's going to start coming in from the other parties in the litigation. Having an IT person who knows the people and issues can only save money and reduce risk. If you can have one person as the "IT liaison," that person can coordinate among the rest of the IT department, the legal staff, and any consultants you may have hired to help, thereby reducing time, billable hours, and headaches.

Finally, there may be some additional modifications to the litigation hold memo. The IT staff should be prepared for last-minute changes, just in case.

In Closing

If you prepare for litigation like any other disaster, you're going to be far better off than if you crossed your fingers and hoped for the best. Knowing what you have, as well as what you don't, and having the ability to retrieve it cheaply and quickly will prevent headaches, save your organization money and time, and make you a hero to the nontechies.

REFERENCES

[1] Federal Rules of Civil Procedure (2006): http://www.law.cornell.edu/rules/frcp/.

[2] FRCP 26(b)(2)(b): http://www.law.cornell.edu/rules/frcp/Rule26.htm.

[3] FRCP 37(b)(2)(A),(B): http://www.law.cornell.edu/rules/frcp/Rule37.htm.

[4] *Colombia Pictures v. Justin Bunnell,* No. CV 06-01093 FMC (C.D. CAL 2007).

OCTAVE ORGERON

an introduction to logical domains

PART 2: INSTALLATION AND CONFIGURATION

Octave Orgeron is a Solaris Systems Engineer and an OpenSolaris Community Leader. Currently working in the financial services industry, he also has experience in the e-commerce, Web hosting, marketing services, and IT technology markets. He specializes in virtualization, provisioning, grid computing, and high availability.

unixconsole@yahoo.com

IN THE AUGUST 2007 ISSUE OF *;LOGIN:*, I explained the Logical Domains (LDoms) technology from Sun and what you can do with it. In this article, I will walk you through the installation process, explaining key requirements for proper installation, as well as suggesting choices you should make during the process.

Prerequisites

For LDoms to function, you will need the correct platform, firmware, OS release, patches, and the Logical Domain Manager software.

Currently, LDoms are only supported on the UltraSPARC T1 (Niagara I) servers, as they are the only UltraSPARC platform with a hypervisor. You can find more information about those servers on Sun's site [1]. In the future, more platforms will be supported as the next-generation Niagara II servers are released.

Each of these servers requires firmware updates to fully support LDoms [2]. The firmware will update and enable the hypervisor software that is contained in the ALOM CMT service processor, which provides the platform lights-out management. In the installation section, you'll find an example of updating the firmware on a Sun Fire T2000.

It is important to have the correct Solaris version to support LDoms. Without the platform and driver support, LDoms will not function properly. The following versions of Solaris are supported:

- Solaris 10 11/06 Update 3 or higher [3]
- Solaris Express Build 57 or higher [4]

Solaris 10 is the commercial version of Solaris and Solaris Express is a preview of Solaris 11 based on the OpenSolaris source code. Solaris 10 should be installed if you require commercial support from both Sun and third-party vendors. However, Solaris Express can be utilized when such requirements are not a concern. Solaris Express provides a preview of developments and features that you will not find in Solaris 10. In this article, Solaris 10 will be utilized. When installing the operating system, it is important to keep in mind that it will become the control domain for the platform.

With Solaris 10, there are patches required to enable full LDoms support. These patches should be downloaded [2] and installed according to the installation instructions included with them.

The last component, the Logical Domain Manager (LDM) software bundle [2], includes the required software packages, installation script, and pointers to online resources.

Installation

Once the operating system and any required patches have been installed, the installation of the firmware and LDM software can begin.

Upgrading the firmware is a multistep process that will require downtime for your server. The first step is to download the corresponding firmware patch for your server [2]. The patch will contain a firmware image file, an installation tool, and some documentation. The installation tool, sysfwdownload, will upload the image to the ALOM CMT service processor. The following example is based on a Sun Fire T2000 running Solaris 10:

```
# unzip 126399-01.zip
# cd 126399-01
# ./sysfwdownload ./Sun_System_Firmware-6_4_4-Sun_Fire_T2000.bin

.......... (10%).......... (20%).......... (30%).......... (40%).......... (51%)
.......... (61%).......... (71%).......... (81%).......... (92%).......... (100%)

Download completed successfully.
```

However, this does not upgrade the firmware. It merely uploads it to the ALOM CMT service processor. To perform the upgrade, you will have to first shut down the server:

```
# shutdown -y -g0 -i 5 now
```

Once the server has shut down, you will have to switch to the ALOM CMT console in order to upgrade the firmware. The console can be reached through the serial port or through the network management port [5]. It is important to ensure that the platform key switch is set to NORMAL to enable the firmware upgrade. Once that is accomplished, the firmware can be upgraded with the flashupdate command:

```
sc> setkeyswitch -y normal
Keyswitch is in the NORMAL position.
sc> flashupdate -s 127.0.0.1

SC Alert: System poweron is disabled.
................................................................................
................................................................................
......

Update complete. Reset device to use new software.

SC Alert: SC firmware was reloaded
sc> resetsc
Are you sure you want to reset the SC [y/n]?  y
```

Once the ALOM CMT reboots, the firmware upgrade is completed. You will notice a change in the versions of the hypervisor, OpenBoot PROM, and the POST diagnostics:

```
sc> showhost
Host flash versions:
   Hypervisor 1.4.1 2007/04/02 16:37
   OBP 4.26.1 2007/04/02 16:26
   POST 4.26.0 2007/03/26 16:45
```

At this point, the system can be powered on and the operating system booted.

Now that the firmware has been updated, it is time to install the LDM software. The software bundle includes the following:

- SUNWldm.v: LDM Software
- SUNWjass: Solaris Security Toolkit (a.k.a. JASS)

The LDM software is fairly small and contained within a single package. It contains the libraries, configuration daemon, command-line interface, SMF service, and man pages for the LDM software.

The Solaris Security Toolkit [6] or JASS is a security-hardening framework. This framework includes configurations that are called drivers. These drivers can disable services, change permissions, lock accounts, enable security features, etc., in a reproducible manner. The toolkit can easily be extended and customized for your environment. It is distributed with other Sun products, such as the management software for E25k, to provide recommended security settings. This is a purely optional component; the LDM software will function without JASS. However, its addition does provide a consistent and flexible security framework.

JASS is included with the LDM software bundle to help secure and harden the primary domain. This is accomplished through the ldm_control-secure driver, which is specifically designed for the primary domain and its services. It will disable all unnecessary services, enable many security features, and lock down access to only SSH.

The LDM software bundle can be installed manually, through Jumpstart, or through the use of the included install-ldm script. This script is included with the software bundle to automate the installation. It will present you with options for hardening the primary domain with JASS. The first option, "a," will install the LDM and JASS software with the driver specifically for the primary domain applied; this is the recommended option. The second option, "b," will only install the LDM and JASS software but will not apply any drivers. The last option, "c," will install the LDM and JASS software but give you the option of selecting a driver to apply. Here is a sample installation session:

```
# Install/install-ldm
Welcome to the LDoms installer.

You are about to install the domain manager package that will enable you to
create, destroy and control other domains on your system. Given the capa-
bilities of the domain manager, you can now change the security configura-
tion of this Solaris instance using the Solaris Security Toolkit.
Select a security profile from this list:
a) Hardened Solaris configuration for LDoms (recommended)
b) Standard Solaris configuration
c) Your custom-defined Solaris security configuration profile
Enter a, b, or c [a]: a
The changes made by selecting this option can be undone through the
Solaris Security Toolkit's undo feature. This can be done with the
'/opt/SUNWjass/bin/jass-execute -u' command.
```

At this point the LDM and JASS software is installed. It is now time to reboot the primary domain.

Configuring the Primary Domain

The primary domain is the first service and the control domain for the platform. Now that all of the prerequisites are installed, it is time to configure the primary domain. The first step is to ensure that the required SMF services are running:

```
# svcs -a | grep ldom
online 18:34:15 svc:/ldoms/ldmd:default
online 18:34:15 svc:/ldoms/vntsd:default
```

The svc:/ldoms/ldmd:default service is responsible for managing the ldmd daemon, which communicates directly with the hypervisor for configuration and management tasks. The svc:/ldoms/vntsd:default service is responsible for providing the virtual network terminal services through the vntsd daemon. If these SMF services are not running, enable them with the svcadm command.

At this point it is good practice to add the following to your $PATH and $MANPATH shell configuration:

```
PATH=$PATH:/opt/SUNWldm/bin
MANPATH=$MANPATH:/opt/SUNWldm/man
```

After all of the prerequisites are installed, all of the resources in the platform are assigned to the primary domain. This can be verified with the ldm command:

```
# ldm list
Name      State    Flags   Cons   VCPU   Memory   Util   Uptime
primary   active   -t-cv   SP     32     32G      0.6%   1h 13m
```

As you can see, all 32 VCPUs and 32 GB of memory are assigned to the primary domain. To enable the creation of other logical domains, resources must be freed and basic services configured. The primary domain should be given at least one CPU core, or 4 VCPUs and 2 to 4 GB of memory:

```
# ldm set-mau 1 primary
# ldm set-vcpu 4 primary
# ldm set-mem 4G primary
```

In this example, a cryptographic thread of a MAU, 4 VCPUs, and 4 GB of memory are assigned to the primary domain. For these settings to take effect, the primary domain must be rebooted. However, before rebooting the primary domain it is good practice to configure the basic services that will support the creation of additional logical domains without causing further reboots.

Creating the virtual console concentrator or VCC service is essential to providing console access to any logical domains created in the future. Only the primary domain can be reached directly via the hardware console; all other logical domains must be reached through the VCC service. When you create the VCC service, a range of TCP ports must be specified. Each of these ports can be bound to one LDom and can be accessed through the telnet command.

```
# ldm add-vcc port-range=5000-5100 primary-vcc0 primary
```

It is important to note that instances of services or devices can be freely named. In the example here, our instance of the VCC service is called "primary-vcc0." The naming conventions used throughout this article take the form of <ldom>-<virtual service or device><instance>.

All virtual storage is serviced by the virtual disk service (VDS). Only one VDS can exist for each service or control domain. This service is created once in the primary domain:

```
# ldm add-vds primary-vds0 primary
```

The last services to be created are the virtual switches (VSWs); these enable logical domains to communicate with the physical network. A VSW should be created for each physical network port on the server. By default the Sun Fire T2000 is equipped with four embedded gigabit Ethernet ports:

```
# ldm add-vsw net-dev=e1000g0 primary-vsw0 primary
# ldm add-vsw net-dev=e1000g1 primary-vsw1 primary
# ldm add-vsw net-dev=e1000g2 primary-vsw2 primary
# ldm add-vsw net-dev=e1000g3 primary-vsw3 primary
```

Once the primary domain resources and services are configured, the configuration must be stored within the ALOM CMT service processor for the hypervisor to reference. This is accomplished by saving the configuration with the ldm add-config <name> command. In this example, I have called my current in-memory configuration "myconfig":

```
# ldm list-config
factory-default [current]

# ldm add-config myconfig

# ldm list-config
factory-default [current]
myconfig [next]
```

This will dump the configurations we have been entering into the ALOM CMT service processor and make it the configuration to use on the next reboot. Now that the current configuration has been saved, the primary domain must be rebooted.

When the primary domain reboots, the configuration will be updated:

```
# ldm list
Name      State    Flags   Cons   VCPU   Memory   Util   Uptime
primary   active   -t-cv   SP     4      4G       0.8%   7m

# psrinfo -vp
The physical processor has 4 virtual processors (0-3)
UltraSPARC-T1 (cpuid 0 clock 1000 Mhz)

# prtdiag -v | grep -i mem
Memory size: 4096 Megabytes
```

You can verify the configuration of the primary domain and the services we created with the ldm list-bindings command. Here is an example of the output reduced to show the key points:

```
# ldm list-bindings
Name:   primary
...
Vcpu:   4
...
Mau:    1
        mau cpuset (0, 1, 2, 3)
Memory: 4G
...
Vds:    primary-vds0
Vcc:    primary-vcc0
          port-range=5000-5100
```

```
Vsw:   primary-vsw0
...
            net-dev=e1000g0
...
Vsw:   primary-vsw1
...
            net-dev=e1000g1
...
Vsw:   primary-vsw2
...
            net-dev=e1000g2
...
Vsw:   primary-vsw3
...
            net-dev=e1000g3
...
```

As you can see, the VCPU, MAU, memory, VDS, VCC, and VSWs are configured. Now that resources and services are available, you can proceed to the configuration of your first guest domain.

Configuring a Guest Domain

Guest domains are consumers of virtual devices and services. As such, these virtual elements must be configured and assigned. Typically, the following resources would be configured:

- VCPU
- MAU
- Memory
- OpenBoot PROM variables
- Storage
- Networking

To begin this process, the guest domain must be created:

```
# ldm add-domain ldom1
```

This will create a guest domain called "ldom1." Resources can now be added to ldom1, starting with VCPU, MAU, and memory resources:

```
# ldm add-vcpu 4 ldom1
# ldm add-mau 1 ldom1
# ldm add-memory 4G ldom1
```

In this example, 4 VCPUs, a MAU, and 4 GB of RAM are allocated. Logical domains only require at minimum one VCPU, which is one of the 32 threads in the Niagara I processor. There is only one MAU thread for each CPU core, of which there are eight total on the Niagara I processor. This can be used to accelerate cryptographic software. Memory can be assigned in varying sizes, from 8K junks to gigabytes at a time.

Each logical domain has its own instance of the OpenBoot PROM (OBP). As such, standard variables can be configured as if it were a stand-alone server. These variables are stored in the hypervisor configuration. These variables can be defined from within the OBP or through the LDM software.

```
# ldm set-variable auto-boot\?=true ldom1
# ldm set-variable local-mac-address\?=true ldom1
# ldm set-variable boot-device=/virtual-devices@100/channel-de-
vices@200/disk@0 ldom1
```

This configures the OBP to auto-boot the logical domain, to configure unique MAC addresses for each network interface, and, finally, to boot off of the first virtual disk. The path defined for the boot disk uses the default device path for all logical domains. The disk target is defined by the last number in the device path.

It's now time to configure storage for your guest domain. There are several options for bootable storage with guest domains:

- Local storage
- SAN storage
- Virtual disk images

Local storage consists of physical disks that are not in use by any other logical domain, including the primary domain. The major limitation in this area is the limited number of physical disk slots on the current Niagara I product line. As such, an external storage array or SAN storage may make more sense.

SAN storage offers greater flexibility and redundancies. It also enables the ability to move logical domains between physical servers in the data center.

Virtual disk images are sparse files the are created with the mkfile command. These files can be virtualized to function as normal storage. This adds another layer of flexibility, since the virtual disk images can be stored locally, on SANs, or on NAS.

In this example, two virtual disk images will be created and added to the VDS service in the primary domain:

```
# mkfile 10g /ldoms/ldom1_vdsk0_10gb.img
# mkfile 10g /ldoms/ldom1_vdsk1_10gb.img
# ldm add-vdsdev /ldoms/ldom1_vdsk0_10gb.img ldom1-vdsk0@
      primary-vds0
# ldm add-vdsdev /ldoms/ldom1_vdsk1_10gb.img ldom1-vdsk1@
      primary-vds0
```

If we had wanted to add a SAN device, the command would look like this:

```
# ldm add-vdsdev /dev/dsk/c6t60060160B5681200944\
2F7677A81DB11d0s2 ldom1-vdsk2@primary-vds0
```

Now that the VDS devices are added, they must be assigned to the guest domain:

```
# ldm add-vdisk ldom1-vdsk0 ldom1-vdsk0@primary-vds0 ldom1
# ldm add-vdisk ldom1-vdsk1 ldom1-vdsk1@primary-vds0 ldom1
```

An important thing to keep in mind is that any virtual storage device bound to a guest domain will appear as if it were a locally attached disk. This removes any additional management layers and simplifies the storage stack for the kernel in the guest domain.

However, it is important to note that virtual storage devices are not presented with SCSI targets to Solaris in guest domains. As such, the device names will be missing the familiar target in the cXtXdXsX standard and appear as cXdXsX. For example, a storage device in the control domain may appear as c4t1d0s0, but in the guest domain it may appear as c0d1s0. The controller number will be determined by the order in which a VDS device was added to the guest domain. There can only be one VDS per service domain. By default the primary domain is the first control and service domain, so all of your disks will be under controller 0. The disk number is determined by the order in which you add the VDSDEV to the guest domain. In our example, ldom1-vdsk0 will appear as c0d0s0 and ldom1-vdsk1 will appear as c0d1s0 in the guest domain. This may affect JumpStart configurations, but it does not affect anything operationally.

Connecting the guest domain to the virtual switches will enable it to communicate with your networks. This involves configuring virtual network ports or VNETs to the VSWs that are connected to your networks. This will configure a unique MAC address automatically and allow access to the connected physical networks and to any other logical domains connected to the same VSW. If we name our VNET instances as ldom1-vnet0 and ldom1-vnet1, we get:

```
# ldm add-vnet ldom1-vnet0 primary-vsw0 ldom1
# ldm add-vnet ldom1-vnet1 primary-vsw2 ldom1
```

These VNET instances will appear, in the order of addition, as vnet0 and vnet1 to Solaris in the guest domain.

Finally, it is time to commit the configuration and start the guest domain:

```
# ldm bind-domain ldom1
# ldm start ldom1

 # ldm list
Name      State    Flags   Cons   VCPU   Memory   Util   Uptime
primary   active   -t-cv   SP     4      4G       0.6%   4h 8m
ldom1     active   -t—     5000   4      4G       0.2%   2m
```

Now you can connect to the virtual console of your guest domain by using telnet and the console number specified under the Cons column from above:

```
# telnet localhost 5000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.

Connecting to console "ldom1" in group "ldom1" ....
Press ~? for control options ..

Sun Fire T200, No Keyboard
Copyright 2007 Sun Microsystems, Inc. All rights reserved.
OpenBoot 4.26.0.build_07, 4096 MB memory available, Serial #66831599.
Ethernet address 0:14:4f:fb:c4:ef, Host ID: 83fbc4ef.

{0} ok
```

The operating system can now be installed in the guest domain through the use of JumpStart. Once this is completed, you'll be able to log in and manage your guest domain:

```
$ ssh ldom1
Password:

ldom1:~ $ uname -a
SunOS ldom1 5.10 Generic_125100-04 sun4v sparc SUNW,Sun-Fire-T200
```

Summary

In this article, you have been given a tutorial on the installation and configuration of logical domains. This tutorial should enable you to explore the use of LDoms and understand the technology. In the next article, I will discuss advanced topics and technology limitations. I will also compare the LDom technology with other virtualization solutions for the Solaris operating system.

REFERENCES

[1] List of Sun servers that support LDoms: http://www.sun.com/servers/index.jsp?cat=CoolThreads%20Servers&tab=3&subcat=UltraSPARC%20T1.

[2] Download site for LDM software, platform firmware, and Solaris patches: http://www.sun.com/servers/coolthreads/ldoms/get.jsp.

[3] Download site for Solaris 10: http://www.sun.com/software/solaris/get.jsp.

[4] Download site for Solaris Express: http://opensolaris.org/os/downloads/.

[5] ALOM CMT service processor documentation: http://docs.sun.com/source/819-7981-11/index.html.

[6] Solaris Security Toolkit (JASS) site: http://www.sun.com/software/security/jass/.

HEMANT SENGAR

IP telephony

BEWARE OF A NEW AND READY-MADE ARMY OF LEGAL BOTS

Hemant Sengar is the co-founder of the vodasec, a voice and data security solution provider to carrier and enterprise networks. His current research interests are in the area of IP telephony and telecommunication network security.

hsengar@vodasec.com

VOICE OVER IP (VOIP), BETTER KNOWN as IP telephony, is aggressively being integrated into the economic and social infrastructure of our lives. But abuse of VoIP will lower people's confidence in this new technology and ultimately hinder its deployment. In this article, I expose a new type of VoIP vulnerability and show how this essential service can be exploited to launch a more potent and stealthy distributed denial-of-service (DDoS) attack affecting both voice and data networks.

IP telephone service providers are moving quickly from low-scale toll bypass deployments to large-scale competitive carrier deployments, thus giving to enterprise networks a choice of supporting a less expensive, single-network solution rather than multiple separate networks. Broadband-based residential customers also switch to IP telephony because of its convenience and cost-effectiveness. In contrast to the traditional telephone system (where the end devices are dumb), the VoIP architecture pushes intelligence toward the end devices (PCs, IP phones, etc.), creating an opportunity for many new services that cannot be envisaged using the traditional telephone system. This flexibility, coupled with the growing number of subscribers, becomes an attractive target to be abused by malicious users.

To break in, attackers may exploit the misconfiguration of devices, the vulnerability of the underlying operating systems, and protocol implementation flaws. Well-known attacks on data networks such as worms, viruses, Trojan horses, and denial-of-service (DoS) attacks can also plague VoIP network devices [1]. Being a time-sensitive service, VoIP is more susceptible to DoS attacks than other regular Internet services. An attacker can disrupt VoIP services by flooding TCP SYN packets, UDP-based RTP packets, or the SIP-based INVITE, REGISTER, etc., messages.

However, if we look at past and current events to identify trends and changes in attacks and targets, then we find that bot-generated DDoS attacks are the most imminent threats to VoIP deployments, as they have been a constant threat to data networks.

The success of bot-generated attacks depends upon two main factors: first, the vastness and diversity of the army of bots, and, second, the bot's distribution over the Internet. Consequently, a bot herder always tries to figure out new ways (such as through worms, Web links, or email attachments) to recruit more hosts into this attacking army. However, there is always a risk of getting caught by law-enforcement agencies for breaking into so many computers. Furthermore, in such digital gang warfare, the rival bot herders may hijack or knock-off these compromised hosts [2], or antivirus scanners may detect and block bot code. But what if, instead of recruiting and compromising new hosts, an attacker finds a ready-made new army of legal bots to launch a more potent and stealthy DDoS attack? This article tries to expose a design error in VoIP services, particularly SIP and the way the INVITE request is used without authentication at the recipient's end of a call. The exploitation of the benign and useful SIP protocol described here deserves our interest for the following reasons: (1) it is new and unexploited; (2) it affects every VoIP telephone, since it is related to the specification rather than the implementation; (3) ironically, many VoIP security devices can also be victimized; (4) it shows a way that a specification can be maliciously exploited.

Background: SIP-based IP Telephony

Session Initiation Protocol (SIP) [3], a standard signaling protocol for VoIP, is appropriately called the "SS7 of future telephony" [4]. It is a text-based application-level protocol to set up, modify, and tear down multimedia sessions with one or more participants. It can also be used to request and deliver presence information as well as instant message sessions. SIP call control uses Session Description Protocol (SDP) for describing multimedia session information. SIP messages can be transmitted over UDP or TCP, but generally UDP is preferred over TCP because of its simplicity and lower transmission delays. However, in some cases, such as the transportation of large SIP messages or the use of TLS, TCP is the only choice.

SIP ARCHITECTURE COMPONENTS

SIP identifies two basic types of components, *user agents* and *SIP servers*. End devices (irrespective of being a softphone or hardphone) are considered *user agents* (UAs). Each UA is a combination of two entities, the *user agent client* (UAC) and the *user agent server* (UAS). The UAC initiates requests, whereas UAS receives requests and sends back responses. Consequently, during a session the UA switches back and forth between a UAC and a UAS. RFC 3261 [3] describes four types of SIP servers, which are implementation-dependent logical entities: *Location Server*, *Redirect Server*, *Registrar Sever,* and *Proxy Server.*

SIP MESSAGES

SIP development is influenced by two widely used Internet protocols: Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP). In SIP, network elements exchange messages as a part of the protocol to set up a call. These messages are classified in two groups: *requests* and *responses*. SIP requests are also called methods and six of them (INVITE, ACK, BYE, CANCEL, REGISTER, and OPTIONS) are described in RFC 3261 [3]. Other methods (in separate RFCs) are also proposed as an extension of the original six methods. Requests are the actions to be taken by UAs or SIP servers. To reply to a request of a UAC, the UAS or SIP server generates a SIP response. Each response message is identified by a numeric status code and, depending upon the range of the numeric status code, there are six different types of responses.

SIP OPERATION

Now we give an example of a typical call setup flow to highlight the usage of SIP request and response messages between user agents UA-A and UA-B. Suppose that the UAs belong to two different domains that each has its own proxy server. UA-A calls UA-B using its SIP phone over the Internet. The outbound proxy server uses the Domain Name System to locate the inbound proxy server of the other domain. After obtaining the IP address of the other proxy server, the outbound proxy server of UA-A sends the INVITE request to the domain of UA-B. The inbound proxy server consults a location service database to find out the current location of UA-B and forwards the INVITE request to UA-B's SIP phone. Exchanging INVITE/200 OK/ACK messages completes the three-way handshake to establish a SIP session [3]. A set of parameters are exchanged via SIP messages (in the message body using SDP) between the two endpoints before an RTP-based voice channel is established. In general, the path of the media packets is independent of that of the SIP signaling messages. At the end of the call, UA-B (or UA-A) hangs up by sending a BYE message. Subsequently, UA-A (or UA-B) terminates the session and sends back a 200 OK response to the BYE message. This example shows the basic functionality of SIP; the detailed description of SIP operation is in RFC 3261 [3].

SIP DEPLOYMENT—PEERING VS. ISLAND-BASED SOLUTIONS

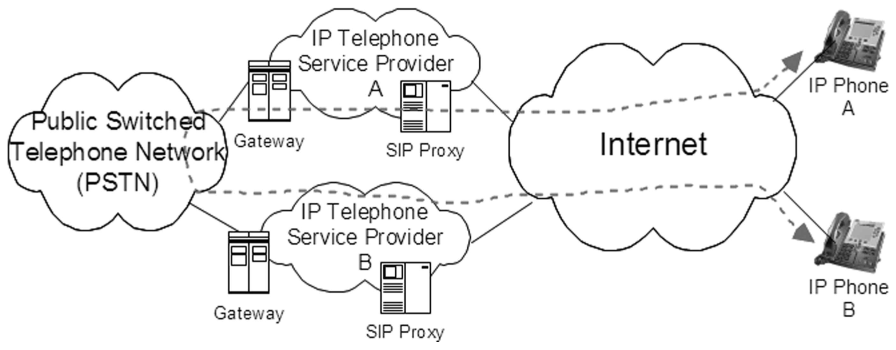

IP Telephone Service Provider A
Public Switched Telephone Network (PSTN)
Gateway
SIP Proxy
Internet
IP Phone A
IP Telephone Service Provider B
Gateway
SIP Proxy
IP Phone B

FIGURE 1: ISLAND-BASED SIP VOIP DEPLOYMENTS

In today's IP telephony world, many of the IP telephone service providers (such as Vonage, AT&T Callvantage, and ViaTalk) operate in a partially closed environment and are connected to each other through the Public Switched Telephone Network (PSTN), as shown in Figure 1. For example, let us assume that user A and user B belong to two different VoIP service providers A and B, respectively. Although the service providers use IP networks to connect with their users, still the calls between users A and B are expected to traverse the PSTN somewhere in the middle. In island-based VoIP deployments the IP traffic is translated into the SS7 traffic [5] (for transportation over the PSTN) and then back into the IP traffic. It is expected that, as VoIP adoption grows, VoIP service providers will interconnect to each other through peering points. Consequently, the calls between any two service providers can be routed through the peering point without traversing the PSTN.

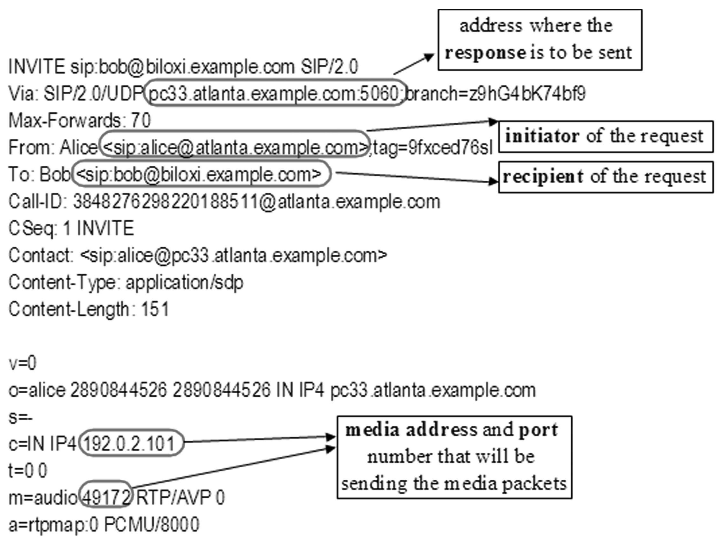
The Threat Model

In a DDoS attack, a number of compromised hosts are used to launch a flooding attack against a particular victim. An attacker installs a daemon on a number of compromised hosts that later on can be requested to start generating spoofed packets directed toward a particular victim target. The enormous number of packets overwhelms the victim's resources, rendering the victim out of service. In the Internet, many network elements such as SIP proxy servers, Web servers, DNS servers, and routers can be defined as *reflectors* because they always respond to some specific type of requests. The attackers can abuse these legitimate and uncompromised reflectors to launch DDoS attacks. The goal of such attacks using reflectors is two-pronged. First, they are used as *stepping-stones*, making such attacks more stealthy so that it is harder to trace back to the actual attacker or real attacking sources. Second, protection becomes difficult, because even if the victimized reflectors are identified, it remains a difficult decision for network administrators to take them out of service, as many legitimate users will also be denied service. However, the use of reflectors is not very lucrative, because a single request generates only one response. Therefore, the number of compromised hosts required to generate spoofed request messages is still large. But what if, instead of a single response, there are a number of response packets for a single request. Such an effect is known as the *amplification* effect. *With the help of reflectors and amplifiers, an attacker can launch a stealthy and more potent DDoS attack using a single machine without possibly compromising any other hosts.*

EXPLOITATION OF THE CALL SETUP REQUEST (INVITE) MESSAGE

Before discussing the exploitation of an INVITE message to achieve both reflection and amplification, we describe its message structure and the purpose of various header fields. As shown in Figure 2, the Via header field contains the address where the caller is expecting to receive the response messages of this request and the From and To header fields contain SIP URIs of the caller and callee, respectively. The Call-ID is a globally unique identifier for this call and the Contact field contains direct route information to reach the caller. The INVITE header fields and message body are separated by a blank line. The session description (media type, codec, sampling rate, etc.) are contained in the INVITE message body. The connection information field (i.e., c=) contains media connection information such as the media's source IP address that will send the media packets. Similarly, the media information field (i.e., m=) contains the media type and the port number.

The exploitation of an INVITE message is based on abusing the *connection information field* contained in the INVITE message body. The SIP proxy server remains at the INVITE header level and routes this message toward the callee without inspecting the message body. The callee's SIP UA parses and interprets the INVITE message and records the media IP address and port number mentioned in the c= and m= fields, respectively. In some cases where the connection information field contains a nonroutable (private) IP address, the SIP UA relies on the *received* parameter of the first Via header field. After a SIP session is established, the callee sends audio packets toward this media IP address and port number. By spoofing the connection information field, an attacker can redirect the media stream toward the spoofed (victim) IP address and port number. In the next section, we give some examples of uncompromised legal bots and the exploitation of the INVITE message.


address where the response is to be sent
initiator of the request
recipient of the request
media address and port number that will be sending the media packets

```
INVITE sip:bob@biloxi.example.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.example.com:5060;branch=z9hG4bK74bf9
Max-Forwards: 70
From: Alice <sip:alice@atlanta.example.com>;tag=9fxced76sl
To: Bob <sip:bob@biloxi.example.com>
Call-ID: 3848276298220188511@atlanta.example.com
CSeq: 1 INVITE
Contact: <sip:alice@pc33.atlanta.example.com>
Content-Type: application/sdp
Content-Length: 151

v=0
o=alice 2890844526 2890844526 IN IP4 pc33.atlanta.example.com
s=-
c=IN IP4 192.0.2.101
t=0 0
m=audio 49172 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

FIGURE 2: STRUCTURE OF AN INVITE MESSAGE

Examples of Legal Bots

CASE I: INTERACTIVE VOICE RESPONSE (IVR) SYSTEM

An interactive voice response (IVR) is a phone technology that allows a computer to detect voice and touch tones using a normal phone call. The IVR system can respond with prerecorded or dynamically generated audio to further direct callers on how to proceed [6]. Both IP and traditional (i.e., PSTN) telephone networks are full of IVR systems, in which a user calling a telephone number is briefly interfaced with an automatic call response system. The typical usage of IVR includes call centers, bank and credit card account information systems, air and rail reservation systems, hospital helplines, and college course registration systems.

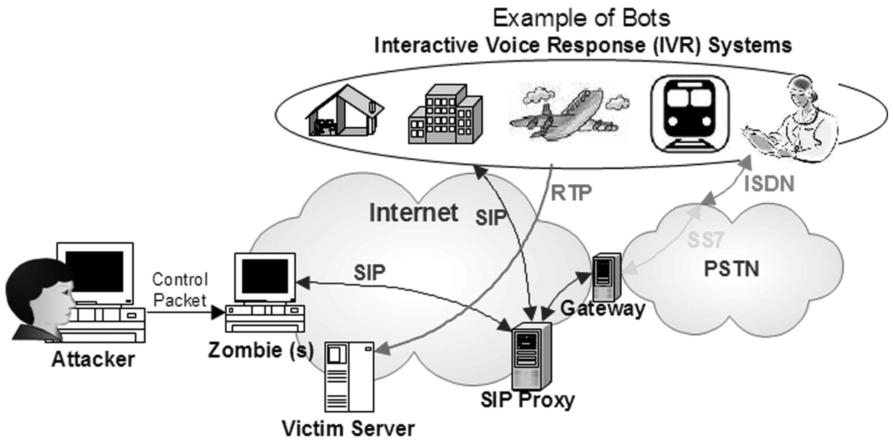


FIGURE 3: IVR SYSTEMS ACTING AS BOTS

As shown in Figure 3, now imagine an attacker, knowing this vulnerability, who sends out a few hundred INVITE messages (while keeping the same media connection address in the message body) to well-known automatic call response systems and establishes fake call sessions with them. In response, the IVR systems flood the victimized connection address with UDP-based RTP packets. In order to establish a call, an exchange of a few call setup SIP messages (i.e., INVITE/200 OK/ACK) can result in a few hundreds to thousands of RTP packets. Such an attack scenario uses both reflection and amplification to make a DDoS attack more potent.

CASE II: USER'S VOICEMAIL SYSTEM

Sometimes when a callee is busy or is not available to answer a phone call, the caller is directed to an answering machine or a voicemail system that plays a greeting message and stores incoming voice messages. An attacker may send fake call requests with the same media connection address to hundreds or thousands of individual telephone subscribers distributed over the Internet. The simultaneous playing of individual greeting messages can overwhelm the link's bandwidth connecting to the victim.

CASE III: USER'S VOICE COMMUNICATION—RTP STREAM

Even if we assume that the callee is not busy and answers a phone call, the callee's voice stream (e.g., Hello, hello . . . or some other initial greeting message) can be directed to a target machine. As in the previous examples, an attacker sends fake call requests with the same media connection address to hundreds or thousands of individual telephone subscribers, and the simultaneous response of subscribers can cause a flooding attack on the victim.

CASE IV: SPIT PREVENTION—THE TURING TEST

In many aspects a voice spam is similar to an email spam. The technical know-how and execution style of email spam can easily be adapted to launch voice spam attacks. For example, first a voice spammer harvests a user's SIP URIs or telephone numbers from the telephone directories or by using spam bots crawling over the Internet. In the second step, a compromised host is used as a SIP client that sends out call setup request messages. Finally, in the third step, the established sessions are played with a prerecorded WAV file. However, voice spam is much more obnoxious and harmful than email spam. The ringing of a telephone at odd times, answering a spam call, phishing attacks, and the inability to filter spam messages from voicemail boxes without listening to each one are time-wasting nuisances.

The Internet Engineering Task Force's informational draft [7] analyzed the problem of voice spam in the SIP environment, examining various possible solutions that have been discussed for solving the email spam problem and considering their applicability to SIP. One such solution is based on the Turing test, which can distinguish computers from humans. In the context of IP telephony, machine-generated automated calls can be blocked by applying an audio Turing test. For example, a call setup request from an unidentified caller is sent to an IVR system where a caller may be asked to answer a few questions or to enter some numbers through the keypad. Successful callers are allowed to go through the SIP proxy server and may also be added to a white list.

VoIP security products such as NEC's VoIP SEAL [8] and Sipera Inc.'s IPCS [9] have implemented audio Turing tests as an important component in their anti-spam product to separate machine-generated automated calls from real individuals. However, an attacker may use these devices as reflectors and amplifiers to launch stealthy and more potent DDoS attacks. For example, to determine the legitimacy of a single spoofed INVITE message, these devices send a few hundred RTP-based audio packets (a 10–20 s audio test) toward the media connection address of an INVITE message. A victimized connection address can be flooded with audio packets if an attacker sends one or two spoofed INVITE messages (with the same media connection address) to several such devices distributed over the Internet.

A Real-World Attack Scenario

To demonstrate a possible DDoS attack, we simulated a real-world attack scenario using IP phones from three different VoIP service providers, namely Vonage, AT&T Callvantage, and ViaTalk. As shown in Figure 4, over the Internet an attacker captures SIP signaling messages exchanged between callers and callees of various VoIP service providers that can later be replayed to launch many different types of DoS attacks toward the subscribers and the SIP proxy server. Most of these attacks are against an individual subscriber, but the INVITE flooding attack can also be launched against a SIP proxy server. However, the media source address spoofing attack discussed in this article is not confined to VoIP systems; rather, it can victimize any voice or data network element.

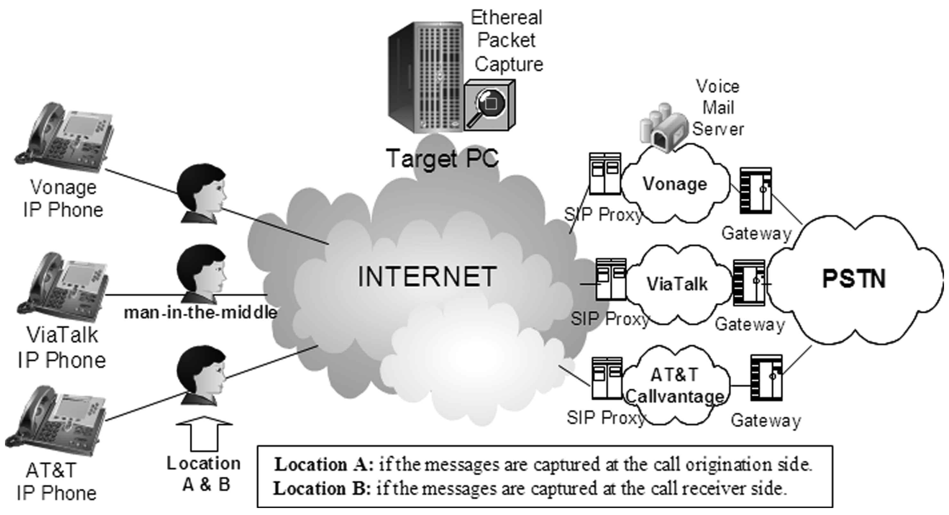


FIGURE 4: REAL-WORLD ATTACK SCENARIO

As shown in Figure 4, the AT&T user talks to both Vonage and ViaTalk customers. The SIP signaling messages exchanged between callers and callees are captured at two locations: *location A* lies between callers and their outbound proxies; similarly, *location B* lies between callees and their inbound proxies. At location A, we observed that in order to prevent *replay* attacks, each of the service providers challenges INVITE messages by sending 401 Unauthorized (in the case of AT&T) or 407 Proxy Authentication required messages that include an MD5 hash of the user's credential and a "nonce" value. This can only be defeated if we have the capability of modifying some header fields (which are not used in MD5 hash computation) and reconstructing the message in real time or by exploiting the implementation of some SIP proxy servers that may accept stale nonce values [10]. However, at location B, there are no such challenge/response messages, leaving the subscribers exposed and vulnerable to abuse. In the sample case study, we exploit the vulnerable and mostly overlooked location B. The captured incoming INVITE messages are reconstructed with a spoofed media address and port number. At a later time, INVITE and ACK signaling messages are replayed while maintaining the same relative order and time. The callee's voice stream (or the playing of the callee's answering machine) is successfully redirected toward the target host.

COULD THIS ATTACK BE PREVENTED?

We now discuss some of the questions that may arise regarding circumvention of the INVITE exploitation attack described in this article. One could argue that the attack can be prevented if the SIP user agent server (UAS) correlates first Via and Contact header fields with the connection address (c=) field of the message body. However, we observe that many services, such as SIP's firewall/NAT traversal and anonymity service, rely on a *media proxy*, thus forbidding the establishment of a correlation between signaling and media destinations.

Conclusion

With the growing acceptance of VoIP and the interconnection between SS7 and IP networks, there is a need to secure both network infrastructures and the protocols used between them. There are many efforts for SIP's implementation vulnerability assessment through syntax testing and test-suite creation. Still, we need to make a thorough revision of the protocol design as well as its intended use. We hope this article will work as a stimulant and bring a concerted effort to prevent any design or implementation flaw that may hinder VoIP deployments or the lowering of IP telephone subscribers' confidence.

REFERENCES

[1] Tipping Point, "Intrusion Prevention: The Future of VoIP Security," white paper, 2005: http://www.tippingpoint.com/solutions_voip.html.

[2] Bob Sullivan, "Virus Gang Warfare Spills onto the Net," April 2007: http://redtape.msnbc.com/2007/04/virus_gang_warf.html.

[3] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, RFC 3261, IETF Network Working Group, 2002.

[4] A.B. Johnston, *SIP: Understanding the Session Initiation Protocol,* 2nd edition (Norwood, MA: Artech House, 2004).

[5] H. Sengar, R. Dantu, D. Wijesekera, and S. Jajodia, "SS7 Over IP: Signaling Interworking Vulnerabilities," *IEEE Network Magazine,* 20(6): 32–41, November 2006.

[6] Wikipedia Encyclopedia, "Interactive Voice Response," April 2007: http://en.wikipedia.org/wiki/Interactive_voice_response.

[7] J. Rosenberg and C. Jennings, "The Session Initiation Protocol (SIP) and Spam—Work in Progress," IETF's SIPPING Group, 2007.

[8] NEC Corporation, "NEC Develops World-Leading Technology to Prevent IP Phone SPAM," product news, 2007: http://www.nec.co.jp/press/en/0701/2602.html.

[9] SIPERA Systems, "Products to Address VoIP Vulnerabilities," April 2007: http://www.sipera.com/index.php?action=products,default.

[10] SIPERA Systems, "Some Implementations of SIP Proxy May Honor Replayed Authentication Credentials," May 2007: http://www.sipera.com/index.php?action=resources,threat_advisory& tid=183&.

DAVID N. BLANK-EDELMAN

practical Perl tools: let me draw you a picture

David N. Blank-Edelman is the Director of Technology at the Northeastern University College of Computer and Information Science and the author of the O'Reilly book *Perl for System Administration*. He has spent the past 20+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and one of the LISA '06 Invited Talks co-chairs.

dnb@ccs.neu.edu

A COUPLE OF YEARS AGO I HAD THE unusual experience of being asked to help design my office when we moved to a new building. Interior design is not something I've ever really dabbled in, but I knew one thing for sure: It had to have as many whiteboards as possible. Readers of this magazine know that I asked for this not out of some fetish for white, slick surfaces. For people like us, drawing often equals thinking. We also know the value of drawing pictures to document infrastructure design, network configurations, data structures, and the lot.

Tools that can make drawing these pictures easier are great. Tools that will actually automate the process are even better. We're going to look at both kinds of tools in this column. I want to introduce you to two of my favorite Perl modules: GraphViz and Graph::Easy.

GraphViz and Graphviz

GraphViz is an easy-to-use Perl module that provides a wrapper around the graph visualization package from AT&T. This package contains a number of programs, which they describe like this:

> The Graphviz layout programs take descriptions of graphs in a simple text language, and make diagrams in several useful formats such as images and SVG for Web pages, Postscript for inclusion in PDF or other documents; or display in an interactive graph browser. (Graphviz also supports GXL, an XML dialect.)
>
> Graphviz has many useful features for concrete diagrams, such as options for colors, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes.

Pedant alert: The name of the AT&T package is Graphviz; the name of the Perl module that acts as a wrapper around Graphviz is called GraphViz (with a capital V). I don't know why the difference in capitalization is there; I can only assume it was an attempt to drive proofreaders batty.

To use the Perl module, you will need to make sure that the Graphviz programs are installed and working on your machine. You can download the source code from http://www.graphviz.org if necessary, but it is pretty likely that there is a Graphviz package available for your operating system through your packaging/installation system of choice (.deb, .rpm, fink/macports, .exe, etc.). From that point on you can choose to ignore the native Graphviz text language (DOT) if you'd like and write only Perl code.

Let's look at the basics of this Perl code because there isn't very much beyond the basics you'll ever need to know to use the module effectively.

The first step is to create a GraphViz object. The creation step is rather important because it is the constructor call (i.e., new()) that determines the format of the graph. This format is passed in via parameters such as layout, as in:

```
my $graph = GraphViz->new( layout => 'neato' );
```

This says that the resulting graph will be processed using the neato algorithm. The Graphviz layout program neato creates spring model graphs (i.e., the ones that consist of balls attached together by lines, mimicking the old molecule-building kits you used in chemistry class). Other layout options include dot (for directed graphs, i.e., trees), twopi (for radial graphs), circo (for circular graphs), and fdp (for spring model graphs like neato but using a different algorithm). The figures in this column are created using the default dot algorithm (i.e., no layout parameter supplied).

By default the GraphViz module will create diagrams with arrows on the lines connecting the shapes on the graph. This can be changed by specifying a "directed" parameter whose value is 0 in the new() call. There are a number of other GraphViz options available in the new() call, so be sure to see the module's (and Graphviz's) documentation.

Once we have a GraphViz object we can start populating the graph. This is quite simple:

```
$graph->add_node('router'); # "router" is the name of that new node
```

If we were to ask GraphViz to create the graph at this point we'd get something quite Zen (see Figure 1).

FIGURE 1

If after years of making this diagram the center of your meditation practice you decide the shape should be a box instead of an oval, you would use this instead:

```
$graph->add_node( 'router', shape => 'box' );
```

Furthermore, if you'd prefer the picture in Figure 2 instead, a third attribute would be specified:

```
$graph->add_node('router', shape => 'box',
                           label => 'Big Blinky Important Thing');
```

Big Blinky Important Thing

FIGURE 2

See the module documentation for other attributes that can be used to change how a node is displayed.

Now that you know how to make all of the shapes you want for your diagram, it is time to connect the dots, err, and nodes. That is done by calling add_edge() for each connection:

```
# add another node first so we have something to connect to
$graph->add_node('web server');
```

```
# connect the node with the name 'router' to the node named 'web server'
$graph->add_edge('router' => 'web server');
```

You probably can guess that there is a panoply of possible optional parameters we can use. For example, if we wanted to label the link between the router and the Web server with its connection type, that would be:

```
$graph->add_edge('router' => 'web server', label => '1000GB-FX');
```

which produces the output shown in Figure 3.

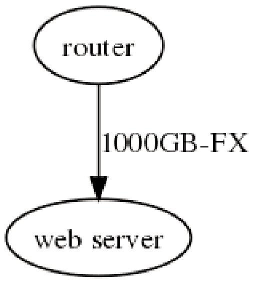


FIGURE 3

Other parameters let us set presentation attributes such as font, arrow size, and color and give hints to Graphviz about how to lay out the resulting graph.

We now know how to make nodes and how to connect them, but we haven't yet seen how to generate a graph that contains those nodes and connections. There are a number of methods that start with as_ for creating the actual graph. For example, as_gif() will create a GIF version, as_png() creates a PNG, as_ps creates Postscript, and so on. GraphViz supports a healthy number of output formats. It can also do neat tricks such as spitting out HTML image map tags.

With the as_* methods it is up to you to decide where the requested output goes. The as_* methods can take filenames, filehandles, references to scalar variables, and even code references if you want to feed the data to a subroutine. If you don't specify an argument it just returns the data, so you can say something like this:

```
print $graph->as_ps;
```

to print the generated Postscript file to stdout.

Congratulations! You have now learned everything you need to know to go off and start making interesting graphs of your own. To help jumpstart your creative process I'll show you one of my examples, and then we'll mention some GraphViz-related modules that can further spark your imagination.

Here's some code that attempts to sniff packets off the Net to show you the connections from hosts on your network to Web servers:

```
use NetPacket::Ethernet qw(:strip);
use NetPacket::IP qw(:strip);
use NetPacket::TCP;
use Net::PcapUtils;
use GraphViz;

my $filt = "port 80 and tcp[13] = 2";
my $dev  = "en1";
my %traffic;    # for recording the src/dst pairs

die "Unable to perform capture:"
    . Net::Pcap::geterr($dev)
    . "\n"
```

```
  if (
  Net::PcapUtils::loop(
    \&grabipandlog,
    DEV        => $dev,
    FILTER     => $filt,
    NUMPACKETS => 50
  )
  );

my $g = new GraphViz;

for ( keys %traffic ) {
  my ( $src, $dest ) = split(/:/);
  $g->add_node($src);
  $g->add_node($dest);
  $g->add_edge( $src => $dest );
}
$g->as_jpeg("fig4.png");


sub grabipandlog {
  my ( $arg, $hdr, $pkt ) = @_;

  my $src = NetPacket::IP->decode( NetPacket::Ethernet::strip($pkt) )
    ->{'src_ip'};

  my $dst = NetPacket::IP->decode( NetPacket::Ethernet::strip($pkt) )
    ->{'dest_ip'};

  $traffic{"$src:$dst"}++;
}
```

First we load up the modules we'll use for network sniffing and dissection plus GraphViz. We set a Berkeley Packet Filter (BPF) filter string to capture SYN packets (i.e., the start of a TCP/IP conversation) to the HTTP port. We set the device for capture and start a capture that will continue until it has received 50 packets. Each time a packet is captured by the filter it will call a subroutine called grabipandlog(). That subroutine takes each packet apart to find the source and destination IP addresses. It then stores a record for each unique source and destination IP address pair encountered.

After the capture has concluded it is a simple to pull all of the connection records out of traffic, adding a node for each source and destination IP address and connecting the two nodes. A graph is generated and written out as a JPEG file. Figure 4 is a simple example of what this program will draw.

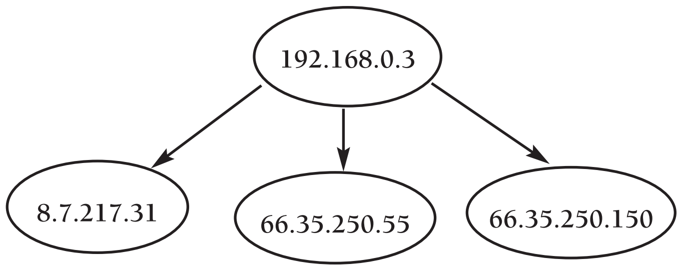


FIGURE 4

If we wanted to, we could make this code a little more complex by:

- thickening or labeling the links between the nodes to indicate amount of traffic or
- showing an internal vs. external Web server distinction by varying the node shapes.

Network traffic diagrams are just one application. The GraphViz module itself comes with several other examples. GraphViz::Data::Grapher and GraphViz::Data::Structure can help you understand complex Perl data structures using two different kinds of graphs. Here's a sample from the GraphViz::Data::Grapher examples directory:

Given the data structure defined this way:

```
@d = ("red",
     { a => [3, 1, 4, 1], b => { q => 'a', w => 'b'}},
     "blue", undef);
```

GraphViz::Data::Grapher will output the graph shown in Figure 5.

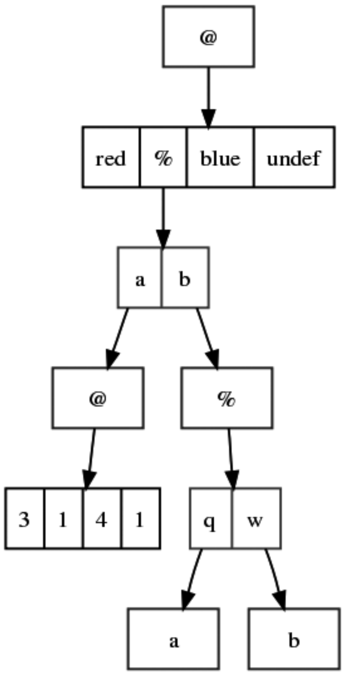


FIGURE 5

Outside of the GraphViz package, there are cool modules to visualize regular expressions, database schema, class diagrams, Makefile structures, parser grammars, XML code, and so on.

Graph::Easy, Baby

I'd like to show you one more package that is similar to GraphViz but is spiffier in a number of ways. Graph::Easy (which is well documented at http://bloodgate.com/perl/graph/manual/index.html) works with a similar idea to that of GraphViz but takes it even further. For example, in addition to writing Perl code like that we've seen for GraphViz, Graph::Easy can input and output data in Graphviz's native format. Being able to output DOT files means Graph::Easy can use Graphviz to create graphs in any graphics format Graphviz supports. Graph::Easy also has a really legible text format it will happily parse to create a graph. Let's look at the Perl and the plain-text method for graph creation.

Here's some sample Perl:

```
use Graph::Easy

my $graph = Graph::Easy->new();
$graph->add_edge ('router', 'web server', '1000GB-FX');
print $graph->as_ascii();
```

This code shows that the general approach for graph specification in Perl is very similar to our previous examples but is a bit more compact. Note that we didn't have to add_node() before creating a connection. We just specified that there was a link between two nodes called "router" and "web server" and that this link should be labeled with "1000GB-FX." Following that specification is a method call not found in GraphViz: as_ascii(). This produces an ASCII drawing like the following:

```
+————————+  1000GB-FX  +———————————+
| router |  ——————————> | web server |
+————————+             +———————————+
```

If you've ever wanted to make an ASCII flowchart for documentation purposes, now you know an easy way to do it.

I could go on and on about the additional graph features Graph::Easy provides (e.g., multiple links between two nodes, links that loop from a node back to itself, links that can point to other links, the ability to create links that fork in two different directions, node groups, more colors and styles, etc.) but I'd like to get to an even more interesting feature I mentioned earlier. Graph::Easy lets you specify graphs using a very easy-to-read text format.

If we wanted to reproduce the simple "two nodes with a link" example that has dogged our every step in this column, we could write:

```
[ router ] — 1000GB-FX —> [ web server ]
```

If we decided the picture made more sense with a bidirectional link, it then becomes:

```
[ router ] <— 1000GB-FX —> [ web server ]
```

You can specify more complicated pictures equally easily. For example, the doc shows this example:

```
[ car ] { shape: edge; }

[ Bonn ] — train —> [ Berlin ] — [ car ] —> [ Ulm ]

[ rented ] —> [ car ]
```

which becomes this picture when as_ascii() is printed:

```
+——————+   train   +————————+         car          +—————+
| Bonn | ————————> | Berlin | —————————————————————> | Ulm |
+——————+           +————————+                      +—————+
                                   ^
                                   |
                                   |
                              +——————————+
                              |  rented  |
                              +——————————+
```

Turning this textual description into a graph for Graph::Easy to generate and output can be done in one of two ways:

- Use the provided graph-easy utility script.
- Ask Graph::Easy to parse the description using Graph::Easy::Parser:

```
use Graph::Easy::Parser;

my $descript = '[ router ] — 1000GB-FX —> [ web server ]';

my $parser = Graph::Easy::Parser->new();
my $graph  = $parser->from_text($descript);

print $graph->as_ascii();
```

Graph::Easy::Parser has a from_file() method if you'd prefer to read the graph description from a file. See the Graph::Easy::Parser doc for more details.

In parting, I think it is important to mention that the ease and power of Graph::Easy hasn't gone unnoticed by other module writers. They've created add-on modules like those mentioned for GraphViz. Here's my favorite example (coincidentally, by the author of Graph::Easy) taken from the module's documentation:

```
use Devel::Graph;

my $grapher = Devel::Graph->new();

my $graph = $grapher->decompose( \'if ($b == 1) { $a = 9; }' );

print $graph->as_ascii();
```

This takes in a piece of Perl code and attempts to generate a flowchart that Graph::Easy can display. Here's the output:

```
################
#   start      #
################
  |
  |
  v
+--------------+
| if ($b == 1) |--+
+--------------+  |
  |               |
  | true          |
  v               |
+--------------+  |
|  $a = 9;     |  | false
+--------------+  |
  |               |
  |               |
  v               |
################  |
#   end        # <+
################
```

Let's end with that pretty picture, drawn just for you. Take care, and I'll see you next time.

DAVID JOSEPHSEN

iVoyeur: opaque brews

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

dave-usenix@skeptech.org

THE JAVA SERVLET CONTAINER MODEL is one of the most popular ways to provide dynamic content to a Web browser. If you haven't had the pleasure of dealing with one, a "Servlet Container" is what you get when you combine a small Web server with a Java program called a servlet. The job of the Web server is simply to accept HTTP requests from the network. The Web server can do things such as parse host names from HTTP headers and perform SSL handshakes as any normal Web server might.

But instead of handling the HTTP requests itself, this Web server passes them to a program called a servlet. The servlet then generates some content, usually in HTML, for the Web server to pass back to its client. The servlet itself is not a binary program but, rather, Java bytecode. It runs inside a Java Virtual Machine (JVM), which provides a runtime environment complete with threading, memory management, much-hyped security, interfaces to other Java and system functions, and presumably everything else a servlet could want. There are quite a few implementations of this model, including Apache Tomcat, BEA Weblogic, IBM WebSphere, and Oracle Application Server, but the basic idea is the same.

In practice it works pretty well until it doesn't, and the complexity introduced by the model makes it nearly as unpopular among system administrators as it is loved by Web developers and their managers. The problem from the systems perspective is the virtual machine. Because the servlet we are trying to troubleshoot or monitor is running inside a virtual machine, our system tools are rendered useless. From the outside, all we can see is the JVM process itself.

If you have a single Web site running on the JVM, then the memory footprint of the JVM is pretty close to the memory footprint of your Web site. Likewise, the CPU load induced by the JVM is pretty close to the CPU load of your Web site. But what if you have five Web sites being vhosted on your Web server? Now a single JVM is running five servlets. Which of them is the one hogging your CPU? Even if you don't have a problem to diagnose, you at least have a capacity planning conundrum, but, believe me, it gets worse.

Let's say you do have a problem and it isn't resource-bound. For example, the application is hanging, or it is crashing outright at unpredictable times. Even with a single servlet in the JVM, tools such as strace, dtrace, and systemtap can only be of limited value, because the JVM has its own internal memory management and thread model. All you can see from the outside is what the JVM's doing, and since it allocates most of the resources it needs up front (including, e.g., its database connection pools), that usually isn't very much.

And speaking of hanging and crashing, I've been privileged in my short career as a Tomcat administrator to see the JVM crash in all sorts of intricate, fascinating, and unpredictable ways. So I can say from personal experience that the servlet container model makes for interesting monitoring fodder in that, short of directly parsing the HTML it returns, it can be difficult to even define a criteria for "functional" that actually describes a functional servlet container. So if you need to monitor specific metrics on your applications, or you've ever wondered just what the heck is happening inside that virtual machine in general, this article will provide some tips, from a systems perspective, for penetrating the black box that is the servlet container.

Profiling

Like their non-Java counterparts, JVM system profilers can provide detailed info on what the JVM is spending its time doing. There are two primary ways of accomplishing this. The first is by registering for messages from built-in instrumentation libraries such as the Java Virtual Machine Tool Interface [1]. The second is by using a process called byte-code injection, wherein known byte-code instructions are detected and preempted with snippets of management-related code. Byte-code injection is more accurate but is much more expensive. Many profilers exist, but most assume that you are a developer operating within an IDE such as Eclipse. For a sysadmin trying to debug a problem on a production system, you can't do much better than hprof [2].

The tool hprof is a simple, powerful JVM profiler that's been included in the JDK since version 5.0. There is nothing to install, and there are no dependencies (well, other than the JVM itself). Simply enable it by passing a -X switch to your JVM options. If you're using Tomcat, for example, you would simply need to add a line similar to:

```
-Xrunhprof[:options]
```

to your startup.sh or Tomcat init file. When the program exits, or whenever the JVM catches a sigquit (kill -3 in UNIX), hprof writes its profiling information to a standard text file. It can provide a stack trace of every live thread in the JVM, heap profiling (what's using all the memory?), and CPU profiling (what's using all the CPU?). It can use either byte-code injection or the Java Virtual Machine Tool Interface, but in practice the former method incurs such a heavy performance penalty that it is, in my experience, unusable for troubleshooting in production environments.

By way of an example, we recently had a problem with several applications hanging on a production Tomcat system. The application hang was accompanied by a CPU spike, so we used hprof to obtain some CPU samples and clicked around the site until the problem showed up. The CPU profile (near the bottom of the hprof dump) looked something like this:

```
CPU SAMPLES BEGIN (total = 206138) Tue Jul 24 22:00:30 2007
 rank     self       accum      count      trace method
    1    42.02%    42.02%       180           481612
oracle.jdbc.driver.OracleDriver$1.<init>
    2    14.94%    14.94%     30796          478299
java.net.SocketInputStream.socketRead0
    3     1.09%     11.09%     22868          495714
java.net.SocketOutputStream.socketWrite0
    4     3.94%     3.94%      8116           495716
java.net.SocketOutputStream.socketWrite0
<snip>
```

The number 1 user of the CPU (at 42%) was the jdbc driver, the glue between our Java application and its Oracle database backend. The number listed under the "trace" column is a unique ID with which we can locate and examine the stack trace of this thread. Toward the top of the hprof dump is the stack trace in question:

```
TRACE 481612:

oracle.jdbc.driver.OracleDriver$1.<init>(OracleDriver.java:1425)
oracle.jdbc.driver.OracleDriver.getSystemProperty(OracleDriver.java:1423)
oracle.jdbc.driver.OracleDriver.connect(OracleDriver.java:840)
            <snip>

oracle.jdbc.pool.OracleDataSource.getConnection(OracleDataSource.java:165)
   pkg.dbgCalls.getConnection(dbgCalls.java:294)
   pkg.dbgCalls.getTrackingId(dbgCalls.java:1843)

   org.apache.jsp.index_jsp._jspService(index_jsp.java:145
```

The last line in the stack trace lists the file and, more specifically, the line number in the file that contains the code that is hogging our CPU. This problem ended up being caused by an ancient copy of the ojdbc14.jar in the WEB-INF folder of the application. Not all problems are this cut-and-dried, and the CPU sampling technique becomes less useful the longer the JVM operates (which makes troubleshooting problems you can't reliably replicate difficult). Also, hprof can't really provide real-time analysis, and it doesn't lend itself to ongoing performance or availability monitoring. Generally, however, hprof is great at providing really specific information like this on demand with a manageable overhead, without raising the vulnerability footprint of the server, and without installing additional software.

JMX

The JVM itself contains instrumentation code for monitoring and management and an API for accessing monitoring and management information in the form of Java Management Extensions (JMX) [3]. JMX is composed of a service that brokers monitoring and management requests to the JVM (called an "mbeans server") and several connectors, which expose the API in various forms such as SNMP and RMI (a Java protocol used by JMX-based monitoring apps). In-house developers can also use the JMX libraries to instrument their applications directly; however, the JVM's instrumentation is sufficient for most monitoring purposes.

If you didn't parse much from that last paragraph, I can sympathize. Java documentation often reminds me of a Frank Herbert book (which is to say, overly concerned with jargon), so I'll attempt another explanation (this time in English). JMX is a direct answer to the problems stated in the opening paragraphs of this article. It provides a window into the operation of the

JVM, and it makes possible such things as a JVM equivalent of the UNIX top program and much more. Further, the same performance and monitoring data can be consumed several different ways, including SNMP and RMI (which makes it possible for a top-like program to monitor a server remotely). When a monitoring vendor says that its app has "Java Integration," that probably means they have a home-brew built-in JMX agent of some sort.

In fact, a JVM equivalent of top called Jtop is included with the current JDK in the demo/management subdirectory. Jtop can give real-time data about the JVM's CPU utilization on a per-thread basis. To use it, you must first enable JMX in your JVM by passing a couple of command-line switches to your startup script. Enabling RMI (remote connections) access to your JVM is a dangerous thing to do. JMX exposes sensitive configuration info such as user names and passwords, as well as the ability to change just about anything related to the operation of the JVM. So you should follow Sun's instructions [4] for enabling authentication and SSL on your JVM's RMI connector. The quick, dirty, and highly insecure way to enable JMX is:

```
 -Dcom.sun.management.jmxremote \
 -Dcom.sun.management.jmxremote.port=1223 \
 -Dcom.sun.management.jmxremote.ssl=false \
 -Dcom.sun.management.jmxremote.authenticate=false \
```

Those switches aren't specific to Jtop; any RMI-based JMX app will require them. Jtop is fairly nifty, but it doesn't even hint at the amount of information that JMX exposes about a running JVM. Jconsole [5], however, included with the JDK and located in JDK_HOME/bin/jconsole, is a fully functional graphical frontend to the JVM. It provides real-time in-depth analysis of a running JVM's memory allocation, CPU utilization, run-time parameters, and stack traces. Jconsole can do everything from providing the exact CPU and memory utilization of a single thread to adding and deleting accounts from the tomcat-users.xml. You do incur quite a bit of overhead using it, but not enough to obviously affect the response time of a running application.

There are several ways to get this info out of JMX and into your monitoring system. The easiest is perhaps JMX's SNMP connector [6]. Simply set the system property for it in the JVM like so:

```
-Dcom.sun.management.snmp.port=portNum
```

Then create an ACL file in JRE_HOME/lib/management/snmp.acl (there's a sample file: JRE_HOME/lib/management/snmp.acl.template). Once this is done, the entirety of JMX is exposed via SNMP for your favorite snmp-enabled monitoring system to make use of. You may even define traps for trap collectors and passive monitoring systems.

For Nagios users, there is a check_jmx [7] plug-in that can query the status of any JMX metric that you see in jconsole. Its syntax is typical of Nagios plug-ins in general. For example, if you were interested in monitoring the JVM's heap you could use:

```
./check_jmx -U \
service:jmx:rmi:///jndi/rmi://myHost:1223/jmxrmi -O \
java.lang:type=Memory -A HeapMemoryUsage -K used -I HeapMemoryUsage -J \
used -vvvv -w 1000000000 -c 1500000000
```

Finally, if you're thinking about rolling your own RMI-based monitoring script, Sun's JMX tutorial [8] has plenty of sample code to get you started, and there's a great white paper on the subject called "JMX Interoperation with non-Java Technologies" [9] on Daniel Fuchs's blog.

JMX is not a panacea. It's still dependent on the JVM itself operating correctly, and that's a pretty big dependency. It does, however, provide excellent insight into the functioning of an operational JVM, and it's allowed me to nip a few problems in the bud before they had the opportunity to destabilize the JVM, and hey, any visibility is a net gain IMHO.

Take it easy.

REFERENCES

[1] http://java.sun.com/j2se/1.5.0/docs/guide/jvmti/index.html.

[2] http://java.sun.com/developer/technicalArticles/Programming/HPROF.html.

[3] http://java.sun.com/j2se/1.5.0/docs/guide/management/overview.html.

[4] http://java.sun.com/j2se/1.5.0/docs/guide/management/agent.html.

[5] http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html.

[6] http://java.sun.com/j2se/1.5.0/docs/guide/management/SNMP.html.

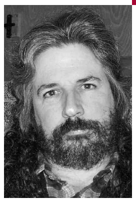
[7] http://www.nagiosexchange.org/Misc.54.0.html?&tx_netnagext_pi1%5Bp_view%5D=808&tx_netnagext_pi1%5Bpage%5D=20%3A10.

[8] http://java.sun.com/j2se/1.5.0/docs/guide/jmx/tutorial/tutorialTOC.html.

[9] http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/JSR262_Interop.pdf.

ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is an information security geek who enjoys surfing (the Internet), sashimi (it makes great bait), and long walks (to the coffee machine and back).

rgferrell@gmail.com

THIS WHOLE DOMAIN NAME SERVICE business has got me to frettin'. Folks have been predicting the imminent catastrophic downfall of the Internet for almost as long as the masses have been aware there was such a thing, but if that vile prognostication ever does come to pass, my money's on DNS being the culprit. Despite some of the vitriolic rhetoric I've seen concerning BIND and other DNS clients, I don't think it's fair to blame them, either, any more than it's fair to blame the jet stream for the untimely demise of your rhododendrons. The real point of failure here is the IP-address-to-name mapping scheme itself.

In the early incarnations of the public Internet, people discovered one by one that if they typed "foowidgets.com" into their browser they often found themselves on the Web page for the Foowidgets company, like as not replete with rotating animated gifs and blinking text (after the advent of Mosaic, anyway). Almost overnight, domain names became precious commodities; people fell all over themselves and anyone in their way in the rush to register names that pertained to their company, product, service, or person.

Once search engines came along and wormed their way into our daily routine, the point of this exercise began to slide inexorably toward mootness (mootitude?). That's not to say that the competition for domain names isn't as fierce as ever, or that people don't continue rightly to despise the bottom-feeding slime creatures who buy up names for which they themselves have no legitimate use, in the hope of reselling them for outrageous profits. They do. It *is* to say, however, that, quite frankly, the DNS system as it currently exists really isn't necessary. (No, that wasn't a misprint: I just had a lot of unused commas lying about that I bought on eBay in a moment of weakness.)

By "really isn't necessary," I mean to say it really isn't necessary, as in, not essential to the proper functioning of the Internet. "How," you may well ask with a hint of incredulity in your voice, "is that possible?" "Why," you indeed will in all probability now further inquire, "would we keep such an elaborate, high-maintenance, high-risk mechanism in place if it weren't vitally important?" The answer to both these questions lies embedded inextricably in the fundamental architecture of the Internet itself.

Pundits like to explain that DNS exists because remembering dotted quad IP addresses is too cumbersome and counterintuitive for humans. Here's why that's just a flat-out dumb statement: IPv4 addresses are 12 digits (of which 4 *must* be <= 2). U.S. domestic phone numbers are 10 digits. I know people who have dozens of phone numbers memorized. How many of you (other than network admins) have a dozen or more IP addresses committed to memory? The fact of the matter is that IP addresses are no more difficult to remember than phone numbers, but no one's ever written a hit song featuring one. IP address assignment and telephone number assignment are very much analogous processes: both involve locating an individual node in a complex multitiered network environment. Using the directory feature on your phone is equivalent to performing a DNS lookup in that it matches a human-friendly name with a network address. This is convenient, yes, but is it vital to contacting that person via phone? Not while we have both paper and electronic phone directories to help us out. In many ways, these tools are like manual search engines for telephone numbers, except you can't zap the pop-up ads as easily.

Search engine algorithms have grown so powerful and the tendrils of their crawler bots so far-reaching that the index page of virtually any domain you could possibly want to visit is already cached in multiple databases totally removed from those used for RFC 1034 (et al.) lookups. It wouldn't take a great deal of tweaking to make these distributed search engine databases authoritative for domain translation. We could also greatly diminish the competition for scarce names and in the process ruin the business models of domain squatters by employing Wikipedia-esque disambiguation pages that allow a large number of people to share one domain name. Forget TLDs: They are instruments of the diabolic and exist primarily to increase the revenue of domain registrars (another business model that will go belly-up). Besides, I'm tired of waiting for ICANN to authorize ".duh" and ".wtf."

How do you go about "registering" a Web presence in this brave new world? Just put it up with the appropriate tags and labels and tell the major search engines about it. As soon as their crawlers index the site, you're on the map. Anyone who types in a search term related to your business will eventually find you, the same way they do now. The only real difference will be that these search terms will be linked directly to an IP address, rather than to a domain name that then must be translated via DNS. You can find a fair amount of this sort of thing already in existence, since not everyone with a Web page has a domain name. You might argue that this gives the search engine companies enormous control over online commerce, but guess what? That's already the case. If you don't believe me, take this simple test:

1. Think of any product or service offered by a company whose name or URL you don't have memorized.
2. Go online and try, without the use of any search engine, to locate someone selling it.
3. Ingest some rich, creamy, artificially flavored trans fat as balm for your humiliating failure.

Your eConsumerism—nay, your very iExistence—is under the thumb of Google, and there ain't much you're willing to do about it, is there?

If the preceding proposal is too radical for you, here's a kinder, gentler alternative. Right now the DNS root zone system is, to borrow Dan Geer's word, *monolithic,* in that it relies on a baker's dozen more or less identical root name servers to point TLD queries in the general direction of the machine authoritative for a given zone. If these root servers are knocked offline or otherwise become unreachable, DNS lookups grind to a screeching halt. But what if we distributed the functions of DNS *at every level* using a BitTorrent-type model, with many, many copies spread across the Internet? The odds against damaging or bringing down all of them at once are considerably higher than in the current failure mode, and as a bonus no single country could claim sole sovereignty over "the Internet." The powers that be claim that's the case now, but I think we all know deep down that it isn't in practice.

Oh, heck: /etc/hosts, anyone?

NICHOLAS M. STOUGHTON

whither C++?

USENIX Standards Liaison

nick@usenix.org

AS I WROTE IN AN EARLIER ARTICLE, revision fever seems to be present in the various standards committees I work with. Along with the revision of POSIX and C (not to mention the possible revision of the Linux Standard Base), the C++ language standard is currently being revised.

To state that this is a big project would be a huge understatement. The working group itself is 3–4 times the size of any other comparable committee, and the scope of the work it has undertaken is on the order of 2^n. It appears a miracle will be needed if the end result is to obtain a clean, complete, and implementable specification in the timeframe they have set themselves.

Among the major things promised for the new version of the language (dubbed "C++0x," since it is hoped that the work will be complete in 2009) are:

- Concurrency support, including:
  - Concurrency memory model
  - Thread-local storage
  - Atomic operations
  - Thread support library
- Programmer Controlled Garbage Collection
- Concepts

The timetable for this project is also extremely aggressive. Officially, the registration ballot, which governs the outline of the document, started earlier this year and has only just completed as I submit this report. The working group members are busier than they have ever been, having held two plenary meetings so far this year, and with another to come in October. At the October meeting, the committee members must decide whether they are done writing the new standard yet or should take another year to complete it. Given the volume of work, it would be amazing if the document was ready to be voted out for final ballot in October, but that is the plan of record.

Concurrency Support

Concurrency support is a topic I've covered in this column in the past. With the advent of multicore chips in even cheap desktop systems, the need to correctly support multiple threads has never been more apparent. What isn't so clear is what should be in the standard once you say "support for concurrency" is to be included. There are plenty of existing threading libraries around, of which, in the C++ space at least, the Boost thread library is probably the most widely used. There are also many applications that use these libraries. What the language standard desperately needs to describe is how the memory model, the low-level atomic data types, and thread local storage work. It is not a major requirement, at least in my mind, to have yet another thread library *per se*. Such a library is proposed to include a new thread-launching API, thread cancellation, mutexes, condition variables, spin-locks, and the rest.

That would mean that all those existing applications would need to be ported to the new standard library. Multithreaded programming is hard enough at the best of times; once you have your program working, you are unlikely to want to rewrite it simply to use the new standardized threads, unless you are forced to (e.g., because you are using a third-party library that does).

One particular area where the current, nonstandard, threading libraries have problems is in handling thread cancellation. If a thread is blocked in an I/O system call (or any blocking system call, for that matter), it is sometimes convenient to signal that thread to tell it to abandon its wait and give up. POSIX has a pthread_cancel interface to do exactly that. Once the canceled thread acts on the cancellation request, it runs its cleanup handlers and dies. The thread cleanup handlers were designed with C++ exception handling in mind.

In the Rationale part of POSIX, thread cancellation is explained well:

> Many existing threads packages have facilities for canceling an operation or canceling a thread. These facilities are used for implementing user requests (such as the CANCEL button in a window-based application), for implementing OR parallelism (for example, telling the other threads to stop working once one thread has found a forced mate in a parallel chess program), or for implementing the ABORT mechanism in Ada.
>
> POSIX programs traditionally have used the signal mechanism combined with either longjmp() or polling to cancel operations. Many POSIX programmers have trouble using these facilities to solve their problems efficiently in a single-threaded process. With the introduction of threads, these solutions become even more difficult to use.
>
> The main issues with implementing a cancellation facility are specifying the operation to be canceled, cleanly releasing any resources allocated to that operation, controlling when the target notices that it has been canceled, and defining the interaction between asynchronous signals and cancellation. . . .
>
> *Cancellation Points*
>
> Cancellation points are points inside of certain functions where a thread has to act on any pending cancellation request when cancelability is enabled, if the function would block. As with checking for signals, operations need only check for pending cancellation requests when the operation is about to block indefinitely.
>
> The idea was considered of allowing implementations to define whether blocking calls such as read() should be cancellation points. It was decided that it would adversely affect the design of conforming applications if blocking calls were not cancellation points because threads could be left blocked in an uncancellable state.
>
> > [from The Institute of Electrical & Electronics Engineers, Inc., and The Open Group, *Draft Standard for Information Technology—Portable Operating System Interface (POSIX®)*, 2007]

The current GCC model is to throw a special sort of exception when a thread is canceled. The exception is similar to typical C++ exceptions, except that it cannot be identified or ignored. The thread unwinds the stack, entering any catch(...) block and destroying objects as needed, until it exits, where it can be joined by another thread that is waiting. The exception is always automatically rethrown as necessary after any catch.

The C++ thread proposal wants to add its own thread-cancellation interface. The proposal at present, however, has nothing to do with thread cancellation as I have just described it (and as everyone is asking for). The proposed mechanism simply requests that the targeted thread throw an exception at the next point it notices the request to do so, and blocking system calls are not mentioned in the list of cancellation points. By calling this thread exception handling mechanism "cancellation," everyone is unhappy. It doesn't interrupt blocked system calls, and it doesn't terminate the thread. Any exception handler can "cancel the cancel" by simply catching and not rethrowing the exception. So it can't be implemented on top of pthread_cancel, and it does not serve any useful purpose beyond some sort of interthread communication mechanism.

But there is so much confusion caused by the terminology that it is proving very hard to come to consensus. There are those who believe that the C++ committee will be a laughing-stock in the community if it publishes a new revision of the standard and it does not have a thread API. I believe that it will be a laughing stock if it does, especially if the revision is anything like the one currently on the table.

There is also the question of mixed C and C++ applications to deal with in this case. Many applications use C language libraries, and there is no guarantee that C++ exceptions will correctly propagate up the stack through C stack frames. Exception-based thread cancellation may run into undefined behavior (e.g., core dumps) in a mixed-language environment. Most modern C compilers do have a mode to enable exception handling (for instance, `gcc -fexceptions`), but there is no guarantee that third-party libraries have been built this way.

This issue threatens to become a major stumbling block for the entire standard. It is possible that the standard may end up including a weaker than necessary thread library that is poorly designed, unimplementable on POSIX, and of little use to anyone. In fact, with this library included, the title of this article might be "Wither C++" instead of "Whither C++."

Garbage Collection

We've all used debugging aids such as Purify or Valgrind to track down memory leaks. In a well-formed C++ program, it is generally easier to avoid some of the more obvious memory leaks because of the way objects are destroyed as they go out of scope, and by use of the exception mechanism.

One thing that C++0x promises to bring to the table is the concept of Smart Pointers. Smart, or Shared, Pointers allow multiple pointers to the same object to exist, reference counting the object referred to via the smart pointer. This prevents the object from being prematurely destroyed, while ensuring that it does get destroyed once the last shared pointer goes out of scope.

However, this in itself doesn't cure all the problems of memory management. One group is trying to add explicit, programmer-controlled garbage collection to the language. There is a partially complete, experimental implementation of this, but little or no real programmer experience in using it.

It does look as if it might be a good debugging aid, to go with those we already have, but it is certainly unclear to many that this highly intrusive feature is worthwhile.

Every single object has a Garbage Collection (GC) attribute: it is either gc_required or gc_forbidden. Most objects shouldn't, in the end, care, in which case they should be gc_safe, which means that it doesn't matter if GC runs on this object or not.

A well-written program will gain nothing from GC (and in fact has to pay a small penalty for it). The real problems come with third-party libraries, which really should be entirely gc_safe. However, it isn't possible to have a program that has a mix of gc_required and gc_forbidden objects. So if a library chooses one of the required/forbidden attributes, the rest of the program (and any other library that is used) must go along with that.

Given the status of the implementation, and the fact that the "standardeze" has only just been written as I write this, it seems unlikely to me that GC will make it in. There remain a host of unanswered issues: Should we have finalizers as well as destructors? (A finalizer is run by the GC to release any nonmemory resources owned by the object.) Does this feature actually provide benefit? Will it lead to better or worse programs? One anecdote used recently related programming experience with GC leading to programs that "flushed the toilet when they noticed the ice-maker in the freezer was empty"; both are devices connected to the plumbing, but with no other obvious connection. Or, in programming terms, "I'm out of file descriptors; let's try garbage collection." GC is *entirely* about memory resources, and nothing else.

Good Stuff

OK, some of the new features in the language are definitely worth waiting for. The revision promises to give us variadic templates, r-value references, constant expressions, better support for generic programming (the "concept" idea currently implemented in ConceptGCC), better operator overloading, improved POD types (leading to better integration with C library functions), explicit virtual functions, strongly typed enums, and most of the library extensions from TR-1 (except the "Special Math" functions, which will move to a new, stand-alone International Standard of their own). If you want to see the details of any of these, go to http://www.open-std.org/jtc1/sc22/wg21/ and look at the individual papers.

POSIX and C++

On a somewhat separate note, several members of the Austin Group (responsible for the development and maintenance of POSIX), together with a good number of the C++ committee members, met independently and looked at the subject of providing a C++ language binding to POSIX. Other languages have done this in the past (notably Ada and Fortran), and it has been helpful. Until now, the answer to how to integrate POSIX facilities into a C++ program has been simply to use the C language libraries that POSIX specifies. However, there are numerous issues with this approach, and several places where POSIX meets C++ in ways that could be (and have been) implemented in a C++ library with C++ type semantics. Thread cancellation is an obvious candidate here, but the idea reaches across the entire POSIX range of functions, including such things as networking, file system access, dynamic libraries, and process control. Until this point, the Austin Group and C++ committee members have been operating as a Study Group under the auspices of IEEE-PASC (the Portable Applications Standards Committee, the original authors of POSIX). This group has agreed to apply for a new IEEE project to develop such a language binding. The official name of the new project is the "POSIX/C++ Language Binding," but I'll refer to it as the "POSIX++" project.

Part of the scope of POSIX++ will be to define the interaction of the C language APIs and any C++ instantiation. For example, what is the interaction between iostreams and file descriptors? What happens if a C library does a `pthread_cancel` on a thread that was created in a C++ module?

It is also within the scope of POSIX++ to define new C APIs to allow for such mixed-language programs and to help C++ library developers implement the new C++0x library on a POSIX platform.

The POSIX++ project is just starting. If you are interested and want to be involved, please feel free to contact me.

book reviews

ELIZABETH ZWICKY,
SAM STOVER, AND RIK FARROW

MANAGE IT! YOUR GUIDE TO MODERN, PRAGMATIC PROJECT MANAGEMENT

Johanna Rothman

The Pragmatic Bookshelf, 2007. 336 pages.
ISBN 978-0-97897392-4-9

This book is about reality-based project management: what you do to try to get a project out the door, with dignity and on a predictable time line. It makes a strong argument that the only way to actually do this is to do some form of incremental scheduling, where you plan only the stuff you actually know about. I'm conflicted about this. On the one hand, I believe it's true. On the other hand, there are still organizations that just don't work this way, and although the book gives advice on coping with and subverting these organizations, I don't think it would have been sufficient for me when I was in one.

Manage It! is full of great advice about the realities of project management. It talks about hallucinating management, the realities of managing physically separated teams, and how to get programmers to provide estimates that mean something. It pushes heavily for down-and-dirty techniques that use low technology to represent what you actually know and that involve gathering real data.

One of my very favorite parts involves actual measurements on how much it costs to fix defects at various stages of a project. Yes, on the projects where she measured, it was more expensive to fix a defect found after release. No, it was not 1000 times as expensive. It was generally more like 32 times as expensive as fixing a defect noticed early on. The numbers were different for different projects, and they didn't go in a straight line. It's a small thing, but it speaks to my need for real data. 32 times is bad enough; you don't need to make up numbers that say that it's 1000 times.

The book is aimed at people who have project management experience. If you're coming at it without experience, particularly if you're at a company with an entrenched culture different from the one the author espouses, you may be bewildered by an alluring but not quite comprehensible description of the promised land of project management. There are appendixes with lifecycle and terminology definitions, but they're probably not going to suffice for somebody who hasn't encountered the terms before. The book is also fairly loosely organized.

I recommend this book for people with some project management experience who want new techniques or to improve their skills. It may also be interesting for new project managers in agile or incremental environments or those who just like jumping into the deep end.

PRACTICAL PACKET ANALYSIS: USING WIRESHARK TO SOLVE REAL-WORLD NETWORK PROBLEMS

Chris Sanders

No Starch Press, 2007. 150 pages.
ISBN 978-1-59327-149-7

It's hard to think of a more powerful tool for network management than a packet sniffer. A good packet sniffer makes all the difference; once you know how to use one, it's like taking a blindfold off. Suddenly you can actually see what's going on! Unfortunately, the other way it's like taking a blindfold off is that when you first do it, you're so overwhelmed with input that you can't understand a thing. Many people never get past that point. There's a crying need for a book that will move such people forward.

If you're the sort of person who used to cheat at games with a hex editor, this book will give you the information you need; it's got a very basic introduction to TCP/IP and higher-level protocols, a good explanation of how to make a machine able to see the packets you need and how to install and use Wireshark (which is the successor to Ethereal), and a quick run-through of some things you can do with it.

The introduction to TCP/IP was written with more practical than theoretical understanding, which is a polite way of saying that, in point of fact, the author doesn't understand what the OSI protocol stack is, although he does a pretty good job of explaining its component parts. He states that it's a recommendation, not a standard. It's not even a recommendation; it's a theoretical model, providing only a description.

The actual packet analyses vary. Most of them look reasonable to me, and they give you some examples of how the tools work. I'm not sure how well they'll translate to further uses for somebody who hasn't done a lot of work with TCP/IP before. One of them is downright wrong; it discusses a traceroute where a router fails to return an acknowledgment. The traceroute slows down at this point, only to pick up to normal speed when it moves to the next hop. The author suggests that this shows that the performance problems on the network are somehow caused by this router. In fact, it shows that the router has an odd ICMP configuration, but there's no reason to believe that this is more than a cosmetic problem with traceroute. It isn't going to affect anything that doesn't do ICMP with the router. It might be a useful clue (for instance, it might point to a situation where all ICMP was disabled, causing problems with path MTU determination); it might also be a complete red herring. There's no way to tell from the information given.

This book makes a nice starting point for somebody who knows nothing about TCP/IP networking and wants to get started with packet analysis. It needs to be supplemented with a good TCP/IP resource if you're really going to get anywhere in the long term.

LINUX SYSTEM ADMINISTRATION

Tom Adelstein and Bill Lubanovic

O'Reilly, 2007. 272 pages.

ISBN 978-0-596-00952-6

I am getting to the point where I am a grizzled old-timer. As such, I get hostile about statements such as "For example, with almost every UNIX distribution, Sendmail is the only choice of mail transfer agent (MTA)." Fifteen seconds of research suffices to tell me that Postfix, for example, is available prepackaged for FreeBSD, NetBSD, OpenBSD, IRIX, Mac OS X, and Solaris and ships with NetBSD. I was still young and optimistic when we started running other mail transfer agents on UNIX. Without getting into long arguments about what's UNIX and what's not, it's safe to say that Linux and UNIX have a strong family resemblance that extends to running pretty much the same applications in most cases.

This resemblance then makes it puzzling that you can cover Linux system administration in 272 pages, when *Essential System Administration* takes 1176. Admittedly, *Essential System Administration* covers both UNIX and Linux, but given that the *Linux System Administration* authors find Linux more complex and capable than UNIX, it ought to take up about half the space, or at least a third. The reason it doesn't is that the authors concentrate on installation instructions for specific software packages. They do provide some explanation of concepts, but not much, and what they do explain is not always right.

For instance, they start right out by having the reader install a bunch of services on an Internet-connected machine, in their default configuration. To do this, you'd have to be a lot more trusting than I'd care to be; it's a big bad Internet out there. Then, they talk about not logging in as root, but using su—so they have you create an admin account, so you can log in as that instead of root. This completely misses the point of not logging in as root, which is to log in as an identified individual user.

Not to obsess about mail, but their discussion of mail transfer agents confuses a mail transfer agent and a mail server. People don't retrieve mail from mail transfer agents. Mail delivery agents don't retrieve mail from mail transfer agents. The mail transfer agent shoves the mail somewhere and leaves it there for the mail delivery agent or the mail user agent to pick up, with no further involvement. Their discussion of open relaying confuses it with spam in general, showing an open relay as allowing inbound spam; the problem with open relays is that they pass spam, which is neither inbound nor outbound but merely passing through.

And it's not just mail; the backup chapter advocates writing your own backup scripts, without discussing any of the risks involved, and then says that databases "have their quirks" when it comes to backups without clarifying what those quirks are. (Their primary quirk is a violent, often fatal, allergy to having files in an inconsistent state, accompanied by behaviors that frequently result in backed-up files being in such a state. Not knowing this will probably get you into nasty trouble.)

If you have a solid conceptual background in system administration but want a leg up installing a Linux system, this is an interesting walk-through of popular alternatives on Debian. If you don't have the conceptual background, it's not very helpful. At least couple it with a serious system administration book.

VIRTUAL HONEYPOTS: FROM BOTNET TRACKING TO INTRUSION DETECTION

Niels Provos and Thorsten Holz

Addison-Wesley Professional, 2007

ISBN 10: 0-321-33632-1; ISBN 13: 978-0-321-33632-3

REVIEWED BY SAM STOVER

This book is so good that I haven't finished it. I've spent so much time actually *doing* the stuff that I haven't even touched a good third of the book. On the one hand, that limits my ability to give a thorough review, but on the other, I feel confident that the bits I haven't read will live up to the part I have read. OK, enough syrup: let me tell you why I like this book so much.

The book begins with almost 20 pages of honeypot and IP background, which I promptly skipped, then went back and read because I have a responsibility to my readership. It's a good thing I did, because beyond the basic IP review, there's a very succinct and appropriate comparison between high- and low-interaction honeypots. This distinction permeates the entire book: the sections are divided between methods and uses for each type. As the names suggest, the differences deal with the complexity and capability of the honeypot: high-interaction systems require more care and feeding but have the potential to collect different data from low-interaction. It's important to note that both types have their uses: they just allow for different applications of honeypot technology.

After the background, Chapter 2 jumps right into high-interaction honeypots and tools such as Q, Sebek, and Argos. Q is an open source virtual machine application very reminiscent of VMware. In fact, the authors walk you through building a virtual machine, using Q, that can be run from the VMware Player. Sebek is basically a rootkit that you install on your honeypot to monitor and collect malicious activity. Argos, though, was the gem of the chapter, in my humble opinion. Argos is a new tool, developed by researchers from Vrije Universiteit Amsterdam, which monitors the honeypot in a way that detects zero-day attacks. Yes, you read that right, zero-day. Argos is a specific kind of virtual honeypot, built using Q, which marks incoming network traffic data, follows it through system memory, and, if a buffer overflow occurs, creates a report and memory dump. Memory dump analysis has been a longtime hobby of mine, and I think this is an excellent example of how that kind of technique can be used to advance the detection of malicious activity. I was impressed enough with this tool that I'm going to try to work it into a future *;login:* article.

After the high-interaction honeypot chapter, the low-interaction honeypot chapter takes you through LaBrea, Tiny Honeypot, the Google Hack Honeypot, and PHP.HoP, which is a "Web-Based Deception Framework." Interestingly enough, neither Honeyd nor Nepenthes is discussed in this chapter, but, luckily for us, each has its own chapter later in the book. I'm a big fan of Nepenthes as low-interaction honeypots go, so I was really happy to see a whole chapter devoted to deploying and managing it. I must admit that I was so busy setting up my Q/Argos setup that I just skimmed over the low-interaction honeypot chapter, but I do plan to go back and spend some time setting up some low-interaction honeypots to see what I can collect. There are definitely some sweet tools in that arena that I want to learn more about. If your interests lie in that direction, there's more than enough in this book to get you started and keep you busy.

It follows logically that if there are high-interaction and low-interaction honeypots, there have to be hybrid systems. Sure enough, Chapter 7 is devoted to tools such as Collapsar, Potemkin, and RolePlayer. Each of these systems tries to balance scalability and capability in a way that gives options to the prospective honeypotter who has needs outside of the strict high- and low-interaction products.

Chapters 8 and 9 deal with honeypots for client-side attacks and honeypot detection, respectively. I think the client-side honeypot is definitely an area of research that needs attention, and this chapter gives a good intro. After all, you can't just expect all the good stuff to happen your way—sometimes you have to go out there and collect it. Chapter 10 gives five different case studies which walk through several different compromises, all of which explore different vectors and targets. Chapter 11 spends some time talking about tracking botnets, and Chapter 12 deals with using CWSandbox to analyze malware.

In all, this book is well written, proofed, and edited. No glaring spelling errors, and the text is concise and to the point. Some of the material, such as installing Argos, is not for beginners, but some of the techniques, such as using Q to build a virtual honeypot, are very accessible to just about anyone. A truly great find: go buy your copy today.

MINIMAL PERL FOR UNIX AND LINUX PEOPLE

Tim Maher

Manning Publications, 2007. 464 pages.

ISBN 1-932394-50-8

REVIEWED BY RIK FARROW

I had wondered just what minimal Perl could be, ever since I noticed a review about it on Slashdot. So I tracked down the publisher and got a copy of my own. I felt that my Perl skills could certainly use some honing, and I would be motivated to read and use this book, as long as it worked well.

Maher has done a fine job providing an alternative path to learning Perl. The "minimal" in the title has to do with Maher's choice in how to present the material, not in the sense of providing the minimal amount of Perl. Maher starts right out by taking the reader to the fictional land of Perlistan, where there appear to be four different languages, but everyone who lives there can understand each other. What he is referring to are different styles used when writing Perl, and his teaching technique is to stick with a single style that is both efficient to use and easier to read for, say, a shell programmer.

The mention of UNIX and Linux people in the title is not gratuitous, as Maher uses comparisons to the Bourne, Bash, and Korn shells and the grep, egrep, find, and awk commands to illustrate how Perl works and how using Perl provides features that you can't get from using the shell and commands alone. I found myself learning about new features of commands (although keep in mind that I learned how to use grep in 1982), as well as other tidbits that may not be new to people who learned Linux/UNIX in the past ten years.

And that's just a side effect of *Minimal Perl*. Maher does a great job of presenting Perl, from command-line arguments, one-liners, to scripts. The first part of the book focuses on comparing Perl features to those you can have with UNIX commands alone, and it works very well at helping people who already know UNIX learn Perl. The second part of the book works with Perl more as a programming language: for example, Chapter 10 is about looping. You might wonder how someone could spend nine chapters and *not* mention looping, but consider just the intricacies of regular expressions and how regular expressions function differently in sed, grep, and Perl, and you will begin to understand the gentle and thorough path Maher takes.

Maher does discuss using modules (early on, in fact) and CPAN. Object-oriented programming barely gets a mention, although the reader gets introduced to the use of objects along the way.

I can recommend the book to people, like myself, who want a thorough refresher course in Perl. I imagine this book will work great for those who have some grounding in UNIX/Linux but don't yet know Perl. In particular, if you know someone to whom you handed the Camel book (*Learning Perl,* by R.L. Schwartz) and he or she just didn't get it, then try *Minimal Perl*. Its simpler approach may provide just the trick needed.

USENIX notes

USENIX MEMBER BENEFITS

Members of the USENIX Association receive the following benefits:

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

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

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

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

DISCOUNTS on registration fees for all USENIX conferences.

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

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

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

USENIX BOARD OF DIRECTORS

CREATING THE EVT WORKSHOP

DAN WALLACH
Rice University, EVT '06 co-chair

We only just completed the second annual USENIX/ACCURATE Electronic Voting Technology Workshop alongside the USENIX Security Symposium, and it's already time to start writing the history of the workshop. My, how time flies.

EVT began when two good ideas collided. Alva Couch, working with the USENIX Board of Directors, wanted USENIX to have a voting workshop, which he originally titled "Verifiable Electronic Voting." This idea found its way to Avi Rubin, who had been working on the issue and was a former USENIX board member. Now, Avi, myself, and a number of other researchers had found out that we were about to be awarded a big grant from the NSF to study electronic voting security. Of course, it was still secret and we couldn't tell anybody until the NSF made the big announcement. So we stalled, knowing full well that one of the things we promised to the NSF was to set up a public workshop on electronic voting. Finally, a few weeks later, NSF made the announcement and we got the workshop rolling.

For readers who have never worked with the USENIX staff, it's hard to appreciate how wonderful it can be to organize a USENIX conference. Ellie Young and her staff really do all the dirty work. As the first chair of EVT, my only responsibilities were managing the program committee and ultimately delivering a list of accepted papers. I didn't have to worry about renting space. I didn't have to worry about catering or registration fees or any of that stuff. (Have I mentioned that the USENIX staff rock?)

So, out went the call for papers, in came the submissions, and off to the races went the reviewers on a compressed timetable. We decided to do away with paper proceedings so we could go cheaper and faster (ah, the delicious irony). In the end, attendees got CD-ROMs and everybody can find the papers online. This year, we had the second annual EVT, and with the luxury of advance publicity, we got many more submissions than we could possibly accept, leading some people to say that we already need *two* voting workshops. And so now I'm working on creating yet another workshop. Details to be announced (and history to be written).

If you have an idea for a new workshop USENIX might be interested in sponsoring, please contact workshopproposals@usenix.org.

2008 USENIX NOMINATING COMMITTEE

ELLIE YOUNG

The biennial elections of the USENIX Board of Directors will be held in early 2008. The USENIX Board has appointed Mike Jones to serve as chair of the Nominating Committee. The composition of this committee and instructions on how to nominate individuals are sent to USENIX members electronically and published on the USENIX Web site.

LETTERS TO THE EDITOR

To the editor:

I enjoyed the article, "Some Lesser-Known Laws of Computer Science" (August 2007), and in that spirit wanted to share my three laws of system design (a term that, for me, includes both hardware and software). I have always been guided by them, but only wrote them down in 1998 when I found that I was repeating them too often to junior programmers who lost sight of why they were doing what they were doing in the first place. (I had thought they were too obvious to require stating, but I guess I was wrong.)

So, without further ado, here are Chessin's three laws of system design (with apologies to the estate of Isaac Asimov):

- First Law: Make life easy for the ultimate end user of our systems.
- Second Law: Make life easy for the ISV, except where to do so conflicts with the First Law.
- Third Law: Make life easy for ourselves, except where to do so conflicts with the First or Second Laws.

—*Steve Chessin*
(steve.chessin@sun.com)

To the editor:

Just read the August 2007 issue, with your "Musings." It seems that *;login:* (and many others) is publishing numerous articles of which the gist is basically "beware."

So, how does a suitably paranoid individual who still wants to get some computing done actually get it done? What are the defensive measures, the procedures, the "safe" software, etc?

I've been doing some on-again off-again research on this topic for a while, and there doesn't seem to be a comprehensive answer. Linux Live CDs are a defense against a large set of attacks, but data continuity (saving you work for tomorrow) is a problem. There are still lots of hazards at the physical and logical layers.

Is there a small set of resources out there that could help the individual computing at home?

—*John Lloyd*
(jal@mdacorporation.com)

John:

The most dangerous activities for home users today are browsing the Web and reading email. I'm not kidding.

The best countermeasure for Web browsing attacks is to segregate your critical use of the Web by visiting key sites (banking, etrading) only from a browser running in a VM. I do this. I even use a Live CD image (Linux). No history, and that is a bother but also a security measure.

As for email, what I do—run a very primitive mail client that cannot execute anything—doesn't work for most people. The problem with email clients is that they will automatically invoke HTML-rendering libraries, and this makes email into a real vulnerability. What other suitably paranoid friends do is use Mac OS X to read their email. I just got back from USENIX Security '07, and noted many people doing this, asked Bill Cheswick just how safe he thought this was, etc. When Leopard comes out, with some real security upgrades, I may do the same thing. But not yet.

—*Rik Farrow*

To the editor:

Reading your August '07 editorial brought a few things to mind that might be of interest to you. First, on labeling URLs as harmful, I recently stumbled across the Netcraft toolbar, http://toolbar.netcraft.com/. The idea is to use the sort of information that Netcraft monitors to rank a Web site's trustworthiness. It also lets them publish Web site rankings based on those using the toolbar. See, e.g., http://toolbar.netcraft.com/stats/topsites?c=IE#65597.

I also recently saw an article about how ineffective these methods of labeling URLs as suspicious are: http://www.rsa.com/rsalabs/cryptobytes/CryptoBytes-Winter07.pdf.

That issue of *CryptoBytes* also contains an interesting article on how easy it can be to guess mother's maiden names.

Later in the article, you commented on how many users one's system has. You mention Apache as a one-user system. Actually, there are two choices here. One is to run Apache (and all the CGI programs) as one user. The other is to use suEXEC to run CGI as different user, making Apache a multi-user system. There are advantages to both systems, depending on your model.

A long time ago, I realised that having lots of "small" users is a good idea. I was trying to track an offensive email sent, and I discovered that it originated from the "nobody" user on a UNIX system. There were a bundle of small services that ran as nobody, and having them all use the same UID meant that typical UNIX tools didn't help track the problem (i.e., files, process accounting, and syslog messages all remember UIDs/usernames). Since then, I try to run each service (and indeed, each piece of each service) as a different UID. That way, if some "nobody" misbehaves, I have a quick way to find the one that's the problem.

Finally, your comment on security and multiprocessing reminded me of a paper by Robert Watson presented at the USENIX WOOT '07 conference, "Exploiting Concurrency Vulnerabilities in System Call Wrappers," viewable at http://www.usenix.org/events/woot07/tech/. It explains how security systems must be careful not to check data that might be modified after the check but before use. Watson demonstrated that this error can lead to exploits of a number of systems.

—*David Malone (dwmalone@maths.tcd.ie)*

David:

The point I was making is that most of our systems are, essentially, single-user systems, especially when you consider that most systems are desktops. And even servers, by default, run as single users, with suEXEC being the exception, not the rule.

—*Rik Farrow*

To the editor:

While I disagree with your review of *Myths of Innovation* (Berkun), in the August '07 issue (I gave the book a negative one elsewhere), that's not the point of this note.

What I'd like to object to is the assigning of the "eureka" moment to Euripides, a Greek tragedian of the 5th century B.C. (?480–?406 B.C.), and depriving Archimedes of Syracuse (~?287–?212 B.C.). As Archimedes has been called "the greatest mathematician of his age," and he did a vast amount of engineering work (see E.T. Bell's *Men of Mathematics* [1937, 4th ed., 1976]), I'm not sure the story, which can be found in Plutarch (~46–127 A.D.), is indeed a myth.

—*Peter H. Salus, Ph.D. (peter@netpedant.com)*

Peter:

Good point: it's Archimedes who should have been credited with the eureka moment—certainly not a Greek playwright.

However, I did enjoy the book. Having worked at many startups, as well as having listened to lots of people who had great ideas that they expected would naturally be accepted and built upon *for the idea's worth alone*, I thought the book was quite important for those people to read.

—*Rik Farrow*

conference reports

THANKS TO OUR SUMMARIZERS

2007 USENIX Annual Technical Conference

Santa Clara, CA
June 17–22, 2007

KEYNOTE ADDRESS

- *The Impact of Virtualization on Computing Systems*

Mendel Rosenblum, Stanford University

Summarized by Francis David (fdavid@uiuc.edu)

Mendel Rosenblum began by describing what virtualization means and identified several important properties that can be attributed to a virtualization system. Hardware is multiplexed or partitioned among multiple virtual machines. Isolation and encapsulation provided by a virtual machine decouple it from the hardware and allow for operations such as suspending and resuming a virtual machine or migrating it from one physical computer to another.

VMWare Workstation now supports some interesting new functionality. It is possible to record and play back execution events in a virtual machine with low overhead, something that may prove invaluable for debugging systems. Virtual Infrastructure is yet another product that supports suspending, migrating, and resuming of virtual machines. Virtualization fuels a changing view of hardware in a data center. Compared to a traditional architecture with individual services tied to individual machines, virtualization enables a virtual server environment where all physical hardware is pooled together, allowing for quick and easy reconfiguration of resources available to a service in response to the load experienced. High availability is achieved because the loss of a physical server can be compensated by running virtual servers on the remaining hardware. Consolidation of virtual machines onto a minimal set of physical machines results in significant energy savings, because the unused physical machines may be powered down. Most of these features are automatically managed by VMware's product and are extremely easy to configure, involving a single check box in a GUI in most cases.

Modern operating systems have evolved to achieve the goal of supporting as many applications as possible while simultaneously providing security, reliability, manageability, and good performance. These driving forces, together with the need for innovation, have resulted in complex operating systems. Virtualization technology has resulted in a change in the view of the role of an operating system. An operating system running in a virtual machine doesn't need complex hardware management and does not have to support a broad range of applications. The OS starts to look like a library that is bundled together with an application, and these virtual machines are termed virtual appliances. This concept is expected to have a big impact on the way software is distributed. Work on the Terra system at Stanford focuses on using different operating systems to support different classes of applications in a virtualized environment. Thus, virtualization provides us with renewed opportunities to improve the efficiency, reliability, and security of system software.

In response to a question about increases in complexity of the virtual machine monitor because of the need to add more device drivers into it, Mendel stated that they expect to build better systems now that they have learned from past mistakes. Also, exploiting new hardware features such as an IO-MMU allows VMWare to reduce the number of drivers that affect the reliability of the system. Andrew Tanenbaum argued that virtualization technology bears significant similarity to microkernels. I asked Mendel whether virtualization has a place in a desktop computer and whether the need for sharing will result in the breakdown of the barriers erected between virtual machines. He replied that the significant sharing among applications today may have been a bad idea to begin with and starting all over again using virtualization may be a reasonable approach going forward. Rik Farrow questioned whether running applications within a VM really provided any additional security to the application and whether using a virtual appliance was the right granularity for a protection domain. Mendel said that in many cases it is not.

TRICKS WITH VIRTUAL MACHINES

Summarized by Francis David (fdavid@uiuc.edu)

- ***Energy Management for Hypervisor-Based Virtual Machines***

  Jan Stoess, Christian Lang, and Frank Bellosa, University of Karlsruhe, Germany

Jan Stoess started the presentation by highlighting the importance of energy management in operating systems. Although there are several existing approaches to OS-directed energy management, modern virtual machine (VM) environments present a unique challenge to energy management because of their distributed and multilayered software stack. A distributed energy accounting scheme is proposed that delegates energy management to a host-level component and a guest-OS-level component.

The host-level energy manager collects energy usage information from all device drivers and tracks energy usage for individual VMs. The guest OS energy manager uses virtualized energy accounting to provide fine-grained application-level energy management. In a prototype implementation using L4 and paravirtualized Linux guests, energy accounting was performed for the CPU and disk. CPU energy consumption was estimated by using processor performance counters for various events. A weighted sum of the various counted events is used to obtain the processor energy consumption. Disk energy accounting uses a time-based approach. Request transfer time is used to directly compute energy usage. Both the CPU and disk energy models also account for idle usage.

Recursive, request-based energy accounting is used to accurately measure the energy spent by each virtual machine. This is important because a virtual device may map to multiple physical devices. For example, energy consumption of a virtual disk is a combination of the energy consumption of the physical disk and the CPU energy consumption of the virtual disk driver. The hypervisor also supports throttling of CPU allocation and shaping of disk requests to regulate energy consumption of virtual machines.

Experiments show that the energy accounting models are quite accurate and that host-level and guest-level enforcement of energy constraints works well. In the future, Jan hopes to support fully virtualized systems and multimodal devices such as multispeed disks.

A member of the audience pointed out that the disk energy accounting model does not consider spin-up and spin-down costs. In response to a question about energy constraints not resulting in fair sharing of time, Jan stated that it probably makes more sense to change scheduling algorithms to provide fair and managed sharing of energy rather than time.

- ***Xenprobes, a Lightweight User-Space Probing Framework for Xen Virtual Machine***

  Nguyen Anh Quynh and Kuniyasu Suzaki, National Institute of Advanced Industrial Science and Technology, Japan

Nguyen Anh Quynh presented research on Xenprobes, a framework for probing inside virtual machines. The probing framework is based on the Xen virtualization system and allows users to register probe handlers for arbitrary instruction addresses in a Xen virtual machine. For example, a handler can be registered for the mkdir system call on a Linux virtual machine to intercept all such system calls on the Linux VM.

Xenprobes exploits the debugging architecture of Xen to inject software breakpoints into the probed VM. Unlike the Kprobes framework, which is tied to Linux, Xenprobes can work with any guest OS supported by Xen. Also, probe handlers can be written in userspace instead of in kernel-space as required by Kprobes. Two types of probes are supported. An XProbe is a pair of handlers that are called just before and just after a probed instruction is executed. It is possible to register null handlers. XrProbe handlers are invoked at the beginning of a function and when it returns. XrProbes allow for examining function call arguments and return values. All probes have full access to VM memory.

The probing process starts with probe registration, which is managed by the framework. Multiple probes at the same address are supported. Probes can be enabled and disabled individually and also from within other probe handlers. All probes are removed when the VM shuts down. Microbenchmarks show that injecting probes into a VM causes a large overhead; the null syscall takes 400 times longer when using an XrProbe and 180 times longer when using an XProbe with one handler. A macrobenchmark that examined the time to decompress the Linux kernel with probes placed in the mkdir, chmod, and open syscalls has more reasonable performance. XProbe and XrProbe experience 6% and 39% increases in execution times, respectively.

A member of the audience pointed out that the code-modifying approach used by the probing framework for breakpoints will not support OS code that checksums itself for verifying integrity.

- ***Virtual Machine Memory Access Tracing with Hypervisor Exclusive Cache***

Pin Lu and Kai Shen, University of Rochester

Pin Lu addressed the issue of estimating the optimal amount of memory that should be allocated to virtual machines (VMs). The hypervisor does not normally have access to information about memory usage inside a VM. In order to obtain information about VM memory usage, Pin advocates that part of the memory that would normally be allocated to the VM be used as a hypervisor cache of VM memory. The cache is designed to contain items that are evicted from the VM memory (exclusive cache). It can be shown that when using LRU for replacement, the hypervisor cache approach does not incur any extra misses when compared to the normal allocation of that memory directly to the VM.

The use of the hypervisor cache enables the use of cache miss ratio curves for direct memory allocation to virtual machines. In particular, this scheme allows the determination of cache miss behavior when using memory sizes that are less than the current allocation. The memory allocation to individual VMs is dynamically adjusted, with the objectives of minimizing the geometric mean of each VM's miss ratio and ensuring that there is a bounded performance loss for each VM when being allocated less memory than its baseline allocation.

The proposed design is not fully transparent, as it requires that the OS notify the hypervisor when there is a page eviction from VM memory. Also, there is some overhead because of page copying and management of the cache. A prototype system was constructed using the Xen virtual machine. Experiments show that the throughput degradation for several non-CPU-bound workloads is less than 20%. Experiments also show that the miss ratio curve prediction is reasonably accurate as well. In a multi-VM experiment, the system correctly reallocates memory to substantially reduce the systemwide page-miss metric.

Pin compared his work with VMWare's ESX server. VMware's ESX server only estimates the working set size and uses that measurement to allocate memory. Experiments show that accessing a large segment of memory sequentially without any reuse causes ESX server to allocate large amounts of memory even though the extra memory allocation does not reduce cache misses.

INVITED TALK

- ***Life Is Not a State-Machine: The Long Road from Research to Production***

Werner Vogels, VP and CTO, Amazon.com

Summarized by Ramakrishna Kotla (kotla@cs.utexas.edu)

Werner Vogels started this talk by introducing himself as a "recovering academic," to emphasize his theme: Why it is hard to take research ideas into production systems, and what can be done about it? His talk mainly focused on issues involved in building and deploying large-scale distributed systems while presenting examples in a wide range of fields to draw analogies.

Werner noted that incremental scalability is the key to building and deploying production-grade large-scale distributed systems such as the one that they have at Amazon.com, which supports seven Web sites with 63 million customers, 1.1 million active sellers, 240,000 Web service developers, 14,000 employees, and 20 fulfillment centers. He defined scalability of a system as the ability to: (1) provide increased performance with increasing resources; (2) provide always-on service; (3) handle heterogeneity; (4) be operationally efficient; (5) be resilient to failures; and (6) be cost-effective as the system grows.

Werner stressed the point that it is very hard to take research ideas into production, presenting many examples and explaining the reasons for this problem. He noted that it usually takes about 20 years before a research idea gets adopted into mainstream systems—for instance, the spreadsheet and the Web. He then explained how hyperlinks and markup languages were developed in the mid-1960s, TCP/IP-based networks came to life in the 1970s, and it wasn't until the mid-1990s that the combination of these three turned into the Internet and Web that we use today.

He then pointed out several important reasons behind the slow adoption of research ideas: (1) making unrealistic assumptions about system characteristics and the environment; (2) not accounting for uncertainty in the real world; (3) multiple differing research approaches in solving a problem.

He pointed out that research in the academic world focuses on the details of the technology itself and is not very focused on the application context of the technology, which results in slow or no adoption of these ideas in production systems. For example, many research approaches assume failures are uncorrelated, whereas correlated failures do happen in reality. Next, he explained that many systems fail to take into account the uncertainty that is present in large-scale systems that are complex and nondeterministic in nature. Finally, he pointed out that having differing and competing approaches in the research community makes it harder for system builders to choose the right approach to solve their problems.

Werner suggested that systems should involve fewer assumptions and explicitly account for uncertainty so that they can be easily adopted and deployed in real-world applications. He suggested using Occam's Razor to select the approach with fewest assumptions when there are competing approaches.

Werner Vogels maintains his personal blog at http://www.allthingsdistributed.com.

NETWORK MONITORING AND MANAGEMENT

Summarized by Andrew Baumann (andrewb@cse.unsw.edu.au)

■ *Hyperion: High Volume Stream Archival for Retrospective Querying*

Peter Desnoyers and Prashant Shenoy, University of Massachusetts

Awarded Best Paper!

Peter Desnoyers presented this paper on Hyperion, a high-performance packet monitor that keeps a packet history, enabling new capabilities in network forensics and management. The primary challenge for the system was handling high packet rates while supporting online querying and indexing and running on commodity hardware.

To support the unusual workload, which required guaranteed write throughput but also included read activity, the authors implemented a specialized filesystem named StreamFS. StreamFS is log-structured but does not use a cleaner; instead, it performs its own data aging simply by overwriting old data as needed. It also interleaves slower and faster tracks on the disk to achieve balanced write speeds.

To support fast querying, Hyperion uses a multilevel index structure based on signatures. Use of the Bloom filter signature ensures that signature queries have no false negatives. Finally, a distributed index layer is provided to distribute summaries of index data to other nodes, allowing for faster queries in a multinode network monitoring system.

Benchmarks show that the full system runs on commodity hardware with negligible packet loss up to 175,000 packets per second, at which point it became CPU-bound. With faster hardware, the authors expect to achieve even higher throughput. The questions that were asked focused mainly on possible optimizations to and uses for the system.

■ *Load Shedding in Network Monitoring Applications*

Pere Barlet-Ros, Technical University of Catalonia; Gianluca Iannaccone, Intel Research Berkeley; Josep Sanjuàs-Cuxart, Diego Amores-López, and Josep Solé-Pareta, Technical University of Catalonia

This paper, presented by Pere Barlet-Ros, covered the problem of how to manage overload in continuous network-monitoring applications. These applications are difficult to construct, owing to the unpredictable nature of network traffic, the increasing processing requirements involved, and the problem of efficiently handling extreme overload situations, which is necessary because overprovisioning is not feasible. The general approach is to shed load when necessary, to get the best possible quality of query results.

The system they have developed extends Intel's CoMo (continuous monitoring), which is a modular passive monitoring system. Modules may perform any amount of processing work. The problem is to manage the CPU usage of the whole system while treating modules as black boxes. The load-shedding scheme automatically finds correlations between network traffic features and the CPU usage of query modules, uses those correlations to predict the CPU load of future queries, and uses that prediction to guide load shedding.

About 50 query-agnostic network traffic features are used; they are lightweight and have a deterministic worst-case computational cost. A linear regression is performed to correlate features to observed CPU load. This regression is expensive to compute, so a feature-selection algorithm is used to remove irrelevant and redundant predictors, resulting in two or three relevant features per query.

Load shedding occurs when the total CPU load predicted for all queries exceeds the available cycles; each query can select packet- or hash-based flow sampling when load shedding is required. Results show that the predictive load-shedding mechanism is much better than the usual reactive load-shedding. Future work includes developing load-shedding techniques for queries that are not robust against sampling, and applying similar techniques to manage memory and disk load. The source code is available from http://loadshedding.ccaba.upc.edu/.

■ *Configuration Management at Massive Scale: System Design and Experience*

William Enck and Patrick McDaniel, Pennsylvania State University; Subhabrata Sen, Panagiotis Sebos, and Sylke Spoerel, AT&T Research; Albert Greenberg, Microsoft Research; Sanjay Rao, Purdue University; William Aiello, University of British Columbia

William Enck presented this work on managing complex router configuration files at massive scale, such as at a large ISP. Precise specifications for individual routers are hard to create, and the level of complexity can be overwhelming. This leads to cut-and-paste errors. Current automated solutions such as network management tools don't account for so-called dirty input data, which can result in a syntactically correct configuration but incorrect operation. The proposed solution is to use a template language to establish consistent device configurations and to follow an iterative approach, where users validate the data inputs. This has been implemented in a tool named PRESTO.

In PRESTO, code snippets for router configuration scripts are defined in active templates, which are close to the system's native configuration language. However, PRESTO is also a framework for data modeling. The information used to evaluate configuration templates is stored in an SQL database; as well as simple value expansion, template code can be repeated for multiple database rows, and database values can also be used as conditions and arguments to functions. The master template is constructed from a series of "configlets," allowing composition of configurations. To manage the dirty data problem, a two-step architecture allows users to first supply the input data and then validate the output.

The PRESTO tool is now in use for many applications, and anecdotal evidence suggests that the two-step process helped to solve the dirty-data problem. Asked whether a better router command language would obviate the need for such a tool, William responded that centralized control and a multistage process to avoid dirty data would both still be necessary, and that the use of templates allowed network architects to select specific optimizations for different situations.

INVITED TALK

■ *Exploiting Online Games*

Gary McGraw, Cigital

Summarized by Jan Stoess (stoess@ira.uka.de)

Gary McGraw described the difficulties of attaining security for online games. Because online games are distributed programs that rely on client-side software, securing them remains a challenging problem. Gary referred to the problem as the "trinity of trouble": online games usually have a permanent connection to the Internet, they are complex entities consisting of massively distributed code, and they typically evolve on the fly in completely unexpected directions.

Gary emphasized, however, that online games act as a bellwether for other distributed systems, since they have a large and growing user community, and since they have become a considerable economic factor. World of Warcraft, the most widely used online game, is typically being played by hundreds of thousands of users simultaneously. With eight million registered users worldwide, the revenue from subscriptions alone already adds up to $1.3 billion, let alone the price to purchase the game itself. There also exists a significant middle market for selling transferable game items such as game weapons or skills. As a result, cheating and breaking online games has an economic aspect, since it allows traders to shortcut the tedious acquisition of valuable items.

Gary then asked whether discussing exploits publicly was actually a bad idea, given that it may spread abusable knowledge, and given that people may suffer from consequences for publicizing exploits, such as Dan Farmer, who was fired by SGI for releasing the SATAN suite. Gary argued that such discussion was still necessary, because otherwise the mechanisms to secure online games would be inefficient. To his own regret, the public is interested mostly in the breaking of systems, less so in their securing.

Gary then explained that current legislation mostly aims at counteracting privacy rather than preventing fraud. However, game-cracking used to be a serious problem and can nowadays easily be prevented via online verification. The current laws are therefore inappropriate to counter online fraud; in fact, game cheating is not even considered illegal. As a result, game companies typically use end-user license agreements (EULAs) with special clauses against fraud. But EULAs are problematic, since no one really reads them, and since many companies use egregious types of EULAs, whose conformity with the laws is more than questionable: Sony's EULA allows a rootkit to be installed on the client; Blizzard allows monitoring by means of spyware; Gator even disallows the removal of the software.

Gary then again illustrated the emerging economic importance of online games: The game market is expected to grow from $6 billion in 2005 to $12 billion by 2010. Games have become a field of study for economists, and there exist exchange rates converting the virtual currency of online games into real currencies. There also exist thriving secondary markets, where game items are sold for prices up to $100,000, and where over half a million Chinese people earn regular income as game item providers, sleeping in cots near the computer between their work shifts.

Gary then explained how game cheats work. There are basically two kinds of techniques, exploits and bots. Exploits leverage game bugs to induce unintended game behavior (e.g., teleporting). Bots perform legal inputs but in an automated fashion, to allow the acquisition of game items that would otherwise require tedious human interaction. Simple bots inject keystrokes and mouse movements to repeatedly perform an action until a certain skill has been achieved. More complex tools directly read or write memory locations, inject cheating functionality into system libraries, or hijack the game's system thread to call internal functions directly. Some state-of-the art tools rely on multiple cores, transparently transferring control from an unmodified game thread to a modified instance on a different core.

Gary outlined the way to overcome the security problems of online games. His approach is founded on three pillars that should govern the design of secure software architectures: a risk management framework, software security touchpoints, and knowledge. Software security touchpoints denote the phases and points during the software design process where developers should think how the system may be exploited. For further reading, Gary referred to his book, *Software Security,* published by Addison-Wesley.

In response to a question about why one would use a dual-core system to exploit a game rather than a kernel debugger, Gary said that programming at user level was less complex than in-kernel development. Another questioner wondered whether virtualization could help in preventing hacking of online games. Gary responded that this may be the case in the beginning, but eventually there will be exploits for virtual machine systems. Finally, Gary was asked where the name "warden" came from in Blizzard. Gary said that it was named like this in the EULA itself.

PROGRAMMING ABSTRACTIONS FOR NETWORK SERVICES

Summarized by Peter J. Desnoyers (pjd@cs.umass.edu)

■ ***Events Can Make Sense***

Maxwell Krohn, MIT CSAIL; Eddie Kohler, University of California, Los Angeles; M. Frans Kaashoek, MIT CSAIL

Maxwell Krohn presented Tame, a new programming model for event-based applications which attempts to combine the (relative) ease of programming associated with thread-based models of concurrency with the power and responsiveness of events. The primary difficulty in event-based programming has been termed "stack ripping" by Adya et al. (USENIX Annual Tech '02) and was illustrated with an example of a simple function with several blocking operations; putting this in event-based form required splitting it into a separate function per blocking point, turning a five-line function into half a screen of code. Other common operations are difficult to express with threads; however, an example of performing multiple blocking operations in parallel was given (with the same function) and required a full screen of bookkeeping code.

Tame is implemented as a source-to-source translator and a set of C++ libraries, and it provides a simple set of primitives for high-performance network programming that are designed to provide the expressiveness and performance of events, combined with the readability of threaded code. The four primitive types are events (`e = event<>;`), wait blocks (`twait{..code..}`), variable closures (`tvars{..decls..}`), and rendezvous objects (`rv = rendezvous<>;`). An event can be triggered (`e.trigger();`), and a `twait` block will execute all the statements within the block and then wait until all events that were created during block execution have been triggered. `tvar` is used to declare heap-based local variables that will survive across calls to `twait`. Finally, rendezvous objects provide finer-grained control over waiting. The example given was maintaining a window of outstanding operations, which could be done in a straightforward fashion using rendezvous.

Results compared Tame with Capriccio, a high-performance thread package, measuring threaded and Tamed versions of a small Web server. Throughput was equivalent, and Tame memory consumption was much lower (3x physical and 5x virtual), owing to its use of a single stack. An informal user study of usability was performed by giving students an assignment where half used Tame and the other half libasync; Tame users averaged 20% fewer lines of source and 50% fewer lines of header files. Tame is in production use in OKWS, an event-based server, and on the okcupid.com commercial Web site.

Tame was described as continuing Adya et al.'s work on threads and events and making practical some ideas from Haskell and Concurrent ML. Other related work included protothreads, Duff's device, and the porch checkpointer. Code is available at www.okws.org, and Eddie Kohler's fork can be found at www.read.cs.ucla.edu/tamer.

Greg Minshall asked about several sources of possible programming errors (functions returning at wait points and long twait blocks). The response was that the first was no worse than the same logic in threads, and the second was bad programming practice. Another question elicited the advice that it is a good idea for caller functions always to create new events to pass to callees. Finally, Oleg Kiselyov pointed out that Tame events were first-class delimited continuations, which was related to some issues preventing exceptions from being used in Tame.

■ ***MapJAX: Data Structure Abstractions for Asynchronous Web Applications***

Daniel S. Myers, Jennifer N. Carlisle, James A. Cowling, and Barbara H. Liskov, MIT CSAIL

Daniel started by explaining how AJAX applications work, and just how dreadful the programming environment is for creating a Web application that responds quickly to user input by fetching server-side data. He then presented MapJAX, a language for this purpose implemented as a small set of Javascript extensions, with source-to-source translation producing pure Javascript. MapJAX provides threads, locks, and a simple map object for accessing server-side data. In addition, for performance MapJAX includes parallel `for` loops and integrated caching and prefetching of map data.

The core language object is a read-only key-value map associated with a server URL, which supplies values to populate the map. Values are retrieved on demand and cached. In addition a user-defined prefetch method can be associated with a map to provide a list of prefetch keys based on request history.

After describing the parallel `for` primitive, `pfor`, Daniel pointed out problems with results being returned in arbitrary order, and the difficulty of preventing this with locks without eliminating parallelism. Instead, MapJAX provides an RLU (reserve/lock/unlock) lock, where each loop of the pfor can reserve a position in the lock queue and only take the lock after the blocking call has returned. After a brief implementation description, the remainder of the talk focused on two test applications that were implemented both with MapJAX and manually with standard AJAX techniques—a search/suggest application such as Google Suggest and a map viewer modeled on Google Maps.

Network latency and several different bandwidth levels (256 to 4096 kb) were provided by dummynet, and latency results were presented. The request/response application was very latency-sensitive, with small requests, and AJAX and MapJAX performances without prefetching were similar. However, prefetching gave the MapJAX implementation a significant benefit at higher bandwidths. The map application was much more bandwidth-intensive (4.8K per image tile). With prefetching, MapJAX performance was similar to AJAX at low bandwidth and showed a slight improvement at 1 mb/s, but increased to a 20:1 difference in latency at 4 mb/s.

Related work includes a number of libraries for AJAX development, although none of these provides thread or locking features. The RLU lock appears to be novel in this work. TAME was also mentioned as being somewhat related. A number of future enhancements are planned: network speed characterization and adaptive prefetching based on this, persistent caching on the client, and mutable server-side data structures.

Geoff Kuenning (Harvey Mudd) expressed the concern that the intriguing RLU lock added another way for programmers to screw something up (locking) that was already hard enough. What if you never act on the reservation? The authors' reply was that a wait/notify mechanism might be possible. Mike Swift (University of Wisconsin) asked whether they had found any painful limitations in Javascript; they hadn't, although there were weird implementation problems such as nested functions being excessively slow.

- ***Sprockets: Safe Extensions for Distributed File Systems***

  Daniel Peek, Edmund B. Nightingale, and Brett D. Higgins, University of Michigan; Puspesh Kumar, IIT Kharagpur; Jason Flinn, University of Michigan

Daniel Peek presented Sprockets, a system for implementing extensions (such as loadable Apache modules) in a safe fashion, unlike other user-space mechanisms, many of which seem to require a tradeoff between performance without safety (using dynamic loading and direct linkage in the same address space) and safety without performance (forking isolated processes).

Sprockets was developed to support extensions to the EnsemBlue distributed file system (e.g., to support camera protocols and MP3 metadata). These can be implemented by extending a user-space server, but safety was essential to avoid corrupting the entire distributed file system. The result is Sprockets, which uses binary rewriting to safely run extensions within the address space of the invoking application. Sprockets has three goals: safety, upgradability, and speed. Extensions should only change state in the core system via return values, and they cannot cause externally visible effects. The system must be easy to add to, and extensions must be easy to implement. Because extensions in EnsemBlue are often invoked very many times, the per-invocation costs should be minimized.

Existing alternatives were contrasted: direct dynamic loading (fast and totally unsafe), fork and exec (difficult to use because of IPC, slow, and do not eliminate safety issues resulting from system calls), and fork without exec (easier to use, but still slow and with syscall safety issues). Sprockets executes extensions in the invoking process address space, with no OS intervention, using binary instrumentation. It uses the Pin tool from Intel and the University of Colorado (rogue.colorado.edu/pin), which dynamically rewrites code and caches it. This allows extension code to safely call core routines, as the core routines will be rewritten and cached when executed as an extension.

Rewriting is used to validate system calls and their arguments, and to trace memory writes to the core application address space, keeping a log of these writes so that the original contents can be restored. To speed up the logging process, inline checks are added to avoid logging the sprocket stack and other known safe areas. Sprockets guards against the following extension bugs: memory leaks/memory stomping (via log/restore), segfault (via signal handler), infinite loops (via timer and signal handler), and mishandled file descriptors (a sprocket can only read/close files it opened, which are automatically closed when it finishes). To keep threads from seeing changes caused by misbehaving sprockets, all application threads are halted before entering a sprocket.

Evaluation was performed by developing and testing three different sprockets, as well as some nonsprocket versions. To test safety, different bugs were injected into the extensions; procedural extensions caused application failure in all cases, fork with no exec did not catch the bad system call, and sprockets caught all of them. Measured execution time for a single extension call was 500 to 750 ns for the extensions tested, contrasted with 1.5–3 ms for the fork/no exec versions of the same functionality. This was considered adequate performance for a system providing strong safety guarantees.

The first question asked about extensions generating code. Pin handles this and will happily rewrite and instrument the generated code. Since Daniel mentioned that Sprockets was not totally safe against malicious extensions, Andrew Baumann (University of New South Wales) asked just what they could do; they could try to undermine a mechanism such as the undo log, and this might be preventable with some additional checks. George Forman (HP) asked whether a copy-on-write address space could be used for multiple sprockets. The authors haven't looked at this but have considered transactional memory.

INVITED TALK

■ ***Second Life***

Rob Lanphier and Mark Lentczner, Linden Lab

Summarized by Xiaoning Ding (dingxn@cse.ohio-state.edu)

The two speakers introduced Second Life (SL) and demonstrated its features using an SL avatar. Then the speakers introduced the architectural changes of the distributed system used by SL. Second Life is a metaverse or a virtual world. The speakers emphasized that SL is not a game, although users may play games in it. SL provides an engine for users to create content in it.

SL is a big system and is still growing rapidly. To illustrate how big the system is, the speakers gave the following numbers. In the system, as many as 100 million SQL queries are made each day, over 45 TB of data have been created by the users, the peak bandwidth reaches 10 Gbps, and 1 PB of traffic is generated each month. The big system supports a tremendous number of activities of a large population. SL has 7.2 million registered accounts, of which about half a million are active residents. More than 60% of residents in SL participate in content creation. This is very different from other platforms. For example, most people read Web pages, but only about 10% of people create Web content. People in SL build items, exchange items, buy and sell items, and have a GDP of 40 million U.S. dollars. SL has its own virtual currency, the Linden dollar (L$), which is exchangeable for U.S. dollars. There are 1.9 billion Linden dollars in circulation. Some people even make real money through the virtual world.

The primitive SL system consists of viewers, simulators, a spaceserver, and a userserver. Viewers comprise client-side software running on users' computers. Simulators are server processes running on machines called sim nodes. Each simulator simulates one geographic region. When an avatar moves to a new region, the sim node may be dynamically changed. In the architecture, spaceserver handles message routings and userserver handles user logins and instant messages. The primitive architecture was improved by adopting a central mysql database and a database server, which performs queries for the simulators. The architecture also introduced an HTTP server and a mail server to increase the functionality of SL. These two servers, together with sim nodes and the userserver, were connected to the central database.

The architecture currently used by SL is similar to the architecture just described. The most important change is to improve the scalability of the database system by clustering and partitioning. Other changes include the use of an individual login server to handle user authentication, etc. In the current architecture, all sim nodes are connected to the databases. When the sim nodes are in remote physical locations from the databases, accessing databases becomes a performance bottleneck. Moreover, a database failure may affect all sim nodes. In the future, the architecture will be changed significantly, so that viewers are connected directly to agent domains and region domains. An agent domain includes the data storage system and computers responsible for handling avatars, including their inventories. A region domain includes the data storage system and computers for simulating the environment and physics of multiple regions. By using the domains, the data and communications are localized and both scalability and availability may be increased.

DISTRIBUTED STORAGE

Summarized by Xiaoning Ding (dingxn@cse.ohio-state.edu)

■ ***SafeStore: A Durable and Practical Storage System***

Ramakrishna Kotla, Lorenzo Alvisi, and Mike Dahlin, The University of Texas at Austin

Awarded Best Paper!

Ramakrishna presented a distributed storage system called SafeStore, which is designed to maintain long-term data durability practically despite a broad range of threats such as hardware and software faults, operator errors, attacks, and disasters. SafeStore maintains data durability by applying fault isolation aggressively. It spreads data redundantly across multiple autonomous storage service providers (SSPs) to prevent data loss caused by physical, geographical, and administrator faults. It restricts the interface between the data owner and the SSPs to guard data stored at SSPs against faults at the data owner site. Outsourcing data storage to SSPs also reduces hardware and administrative costs by exploiting economies of scale. The challenge of using SSPs is that the users have only limited or no control over SSPs. To achieve high end-to-end durability practically, SafeStore uses the following techniques.

First, it spreads data efficiently across autonomous SSPs with an informed hierarchical coding scheme. SafeStore uses both inter-SSP and intra-SSP redundancy; that is, it first stores data redundantly across different SSPs and then each SSP replicates data internally. The informed hierarchical coding scheme deals with the problem of distributing overall redundancy optimally between inter-SSP and intra-SSP redundancies. Under the scheme it is assumed that each SSP exposes a set of redundancy factors it supports and the data owner can control intra-SSP redundancies by choosing a factor. The scheme first maximizes inter-SSP redundancy heuristically, by choosing minimal intra-SSP redundancies. Then it distributes the remaining redundancies uniformly across intra-SSP redundancies. The heuristic is based on the intuition that the durability depends mainly on maximizing inter-SSP redundancy and is only slightly affected by intra-SSP redundancies. The informed hierarchical coding scheme can achieve close to optimal durability.

Second, SafeStore performs an efficient end-to-end audit mechanism on SSPs. The audit mechanism quickly detects data losses at SSPs, so that data can be recovered before unrecoverable loss happens. In the audit mechanism, the auditor sends the SSP a list of object IDs and a random challenge. The SSP computes a cryptographic hash on both the challenge and the data and then sends the information back. The auditor randomly spot-checks a portion of the objects by retrieving the corresponding data and verifying the cryptographic hash. SafeStore classifies SSPs into three groups: honest and active SSPs, which verify data integrity proactively and notify data losses; honest and passive SSPs, which rely on audit to check data integrity; and dishonest SSPs, which may lie even if data is lost. Regular audits detect data losses quickly with passive SSPs and spot-checks detect dishonest SSPs with high probability. The audit mechanism is cost-effective because most of the audit work is offloaded to SSPs and there are only occasional spot-checks that need data transfers for a small subset of objects. The audit mechanism provides two 9s or better durability.

The evaluation shows that SafeStore can provide high durability practically and economically, with cost, availability, and performance competitive with traditional systems. More information can be found at http://www.cs.utexas.edu/~kotla/SafeStore.

- ***POTSHARDS: Secure Long-Term Storage Without Encryption***

  Mark W. Storer, Kevin M. Greenan, and Ethan L. Miller, University of California, Santa Cruz; Kaladhar Voruganti, Network Appliance

Mark W. Storer presented a secure, long-term storage system called POTSHARDS. POTSHARDS provides security by secret splitting, instead of encryption, which is unsuitable for long-term storage because of the difficulties of key management and updating cryptosystems over long time periods.

To store data, POTSHARDS breaks the data into *n* pieces, called shards, in a way that *m* out of *n* shards must be obtained to recover the data and any set of fewer than *m* shards contains no information about the data. Then POTSHARDS returns the shard IDs to the user and spreads the shards across multiple independently managed archives. The user maintains a private index that maps the data to shards for fast retrievals. Keeping the index private reduces the threat of inside attackers. To retrieve data, the user provides shard IDs and authentication information. POTSHARDS requests *m* shards from different archives and rebuilds the data from the shards. Thus the security is enforced by hiding shard locations and performing authentication on multiple independent archives.

POTSHARDS uses approximate pointers to recover data even if all indices of a user's data have been lost. Approximate pointers point to multiple "candidate" shards that might be the next that can be used to reconstruct the data. Approximate pointers provide sufficient clues for users to recover their data by trying only candidates they own. Meanwhile, the approximate pointers greatly increase the workload of intruders, because the intruders have to try all the candidates. POTSHARDS also uses a secure distributed RAID technique to provide availability and data recovery.

The performance evaluation on a prototype implementation shows that POTSHARDS can write data at 3 MB/s and can read data at 5 MB/s, which are acceptable rates for archiving. The throughputs were tested with 12 clients. With more clients, POTSHARDS can achieve higher throughputs.

- ***Dandelion: Cooperative Content Distribution with Robust Incentives***

  Michael Sirivianos, Jong Han Park, Xiaowei Yang, and Stanislaw Jarecki, University of California, Irvine

Michael Sirivianos presented the work of Dandelion, a system designed to provide robust incentives for cooperation in commercial P2P content distribution with sufficient scalability. The speaker emphasized that Dandelion is not a distributed storage system.

Although the popular BitTorrent protocol has incorporated tit-for-tat incentives, it does not encourage seeding and still allows modified clients to free-ride. With robust incentives, Dandelion increases the network's aggregate upload bandwidth by motivating clients to upload even when they are not interested in the network's content and by preventing free-riding.

Dandelion builds a virtual credit system, in which a client honestly uploading to its peer is rewarded by virtual credit and a client obtaining the correct content is charged the same amount of credit. In the system, credit can be redeemed for rewards such as discounts or monetary awards; thus, clients are always encouraged to upload. The system prevents free-riding because the only way a client can obtain valid content and can earn credit is by paying credit or uploading valid content, respectively. To prevent client cheating, Dandelion uses a cryptographic fair exchange scheme based on symmetric key cryptography, in which the server acts as the trusted third party mediating the content exchanges. Thus, Dandelion trades scalability for robust incentives. To prevent Sybil attacks, Dandelion maintains only authenticated paid accounts.

Michael Sirivianos and his team implemented a prototype of Dandelion and evaluated it on PlanetLab. With a server running on normal commodity hardware (dual Pentium D 2.8-GHz CPU and 1 GB RAM) with a moderate network bandwidth (1 to 5 Mbps), Dandelion can support about 3000 clients downloading 256KB chunks with a rate of 256KB/s. When the network bandwidth is low (less than 4 Mbps), the bandwidth is the bottleneck. When the bandwidth is high (larger than 5 Mbps), CPU is the bottleneck. The evaluation also shows that robust incentives for seeding substantially improve downloading times, especially for small files, and the performance of Dandelion clients is comparable to those in BitTorrent.

One member of the audience asked what the differences between Dandelion's and eMule's incentive schemes are. The presenter responded that eMule peers maintain pairwise credit balances and give high service priority to peers with high credit. This in essence is a tit-for-tat scheme and has the same weaknesses as BitTorrent's TFT, i.e., it provides no incentives for seeding, and it is susceptible to free-riding.

INVITED TALK

- ***Specializing General-Purpose Computing: A New Approach to Designing Clusters for High-Performance Technical Computing***

Win Treese, SiCortex Inc.

Summarized by Jan Stoess (stoess@ira.uka.de)

Win Treese presented SiCortex's new approach to designing clusters for high-performance computing. Supercomputers are often based on specialized hardware and programming environments, which are expensive and obstructive to rapid innovation. General-purpose computing, in turn, uses standard software and commodity PCs and shows an amazing technology curve. However, commodity PCs are optimized for desktop and server systems rather than for the specific demands of technical computing. The challenge is therefore to unite the best of both worlds: to build a supercomputer that uses general-purpose hardware and a standard programming environment but that is also specifically designed for technical supercomputing.

Win presented a brief sketch of the history of supercomputing: supercomputers such as Cray, CM1, or BlueGene are all expensive machines, each with a different programming environment. They place high demands on the skills of their users and maintainers. Also, supercomputing companies tend to have a hard time generating any income. An alternative and cheaper model is therefore to take lots of cheap PCs and cluster them using commodity interconnect technology such as Ethernet—as was done in the Beowulf project. Even with optimized interconnects such as Myrinet or Infiniband, PC clusters are still cheaper than their big, specialized brothers.

Typical workloads of supercomputers are climate simulations, mechanical design problems, or life science simulations. The applications usually run for weeks and consume every available cycle. They tend to operate on huge data sets in a cache-unfriendly manner, and they place high demands on the communication infrastructure. Programs are usually written in Fortran, a language that permits many optimizations to be made during compilation. More recently, Java and Python have become popular as well, mostly because of the benefits in programming productivity.

By now, HPTC computing on PC clusters has become a mainstream phenomenon. Sales of Linux cluster hardware have reached $6 billion in 2006. The advantages of PC clusters are their cheap prices for hardware, software, and interconnects; their support for emerging software standards such as Linux, MPI, or Fortran; finally, their amazing technology curve.

However, many challenges remain for PC-based supercomputing. Because PCs were originally designed for personal computing, PC clusters often show little computational efficiency, have high power consumption, and generate a lot of heat. Also, the parts are likely to fail and the interconnect is slow. However, software development and standards play a significant role in the overall costs of a supercomputer. A potential way out is, therefore, the replacement of commodity hardware with a specialized system that still retains support for the standard cluster programming environment.

Win then explained how SiCortex has built such a system. They aimed to realize a 1,000-node cluster with near-microsecond communication delay in a cabinet-sized box. Their main design principle was "logic of power." Lower power consumption results in less heat, which, in turn, allows components to be located closer together and interconnect wires to be shorter. Reduced heat also increases the reliability, and reduced power consumption saves energy during memory stalls. Win introduced two supercomputers presently manufactured by SiCortex. The big model, the SC5832, has a floating-point capacity of 5832 gigaflops, 972 6-core nodes at 500 MHz each, 7.8 GB of memory, and roughly 2,900 interconnects. It consumes about 18 KW and fits into a 5x5x6–foot cabinet. The smaller model, with 648 gigaflops, 108 nodes, and 2 KW power consumption, fits into a half-width standard 19-inch rack. The software platform consists of a standard, open-source Linux environment with support for GCC, Fortran, MPI, and even Emacs. SiCortex also provides an integrated Lustre file system that can be used for direct-connect storage, for external storage servers, or even for RAM-based file systems.

Win concluded with the observation that general computing techniques and specialized knowledge on supercomputing applications can be mixed very well to support powerful and usable high-performance technical supercomputing.

Someone asked about the price of the systems, and Win gave ballpark numbers of $200,000 for the small box and $1.5 to $2 million for the big one. Another question was how nodes could be replaced on a failure. Win stated that the hardware was hot-swappable, but the system still lacked support for software hot-swapping. He noted, however, that software hot-swapping would be straightforward to implement. Asked how software could be ported to the supercomputers, Win responded that single-CPU programs would need redevelopment to exploit parallelism; other programs can simply be recompiled for MIPS targets. Win was asked how the system deals with multicast or broadcast messages. He explained that the DMA engines provide hardware commands to build up broadcast trees efficiently. Finally, a questioner inquired how jobs are scheduled in their system. Win said the system ships with a job scheduler developed by Lawrence Livermore National Laboratory.

DATA AND INDEXING

Summarized by Peter J. Desnoyers (pjd@cs.umass.edu)

■ ***Using Provenance to Aid in Personal File Search***

Sam Shah, University of Michigan; Craig A.N. Soules, HP Labs; Gregory R. Ganger, Carnegie Mellon University; Brian D. Noble, University of Michigan

Sam Shah presented a system that tracks causal relationships between files in a system to improve desktop search results. In the first part of the talk he described the approach taken in this work, comparing it with previous work; the rest of the talk focused on results from a rigorous usability study with test subjects. Google Desktop and others use static indexing based solely on file contents; Soules and Ganger [SOSP '05], however, have shown that dynamic information (patterns of use) can be used to reorder search results and give user-perceived improvement in search performance. In particular, provenance information—indications of which files were referenced to create another file—can be used to infer relations between files.

The current state of the art for this uses temporal locality—that is, if read(A) is followed by write(B) within a T-second time window, then A and B are related. This captures reading an email and then copying it into a text document; however, it fails in a number of cases. For example, one test user kept a CAD application open all day; using temporal locality inferred relationships between the CAD files and all other user activities.

The authors instead use causal relationships—A and B are associated if read(A) and write(B) occur in the same process, or in two processes with an IPC path during the intervening interval. Each approach handles some cases better than others. These relationships are used to create a DAG (Directed Acyclic Graph) with edges weighted by the number of relationship events seen, and then a breadth-first traversal method is used to redistribute relevance weights returned by static search, resulting in a new weighting used to order results for the user. Measurement of both causal and temporal relationships was implemented on win32, using binary rewriting to interpose between applications and the file system. The search application was based on Google Desktop, using the GDI API with provenance-based reordering of results.

Shah discussed why a typical corpus-based approach to measuring precision and recall, as used in many information retrieval studies, was not appropriate to this application. Instead, a real user study over a period of time was needed. A full randomized controlled trial would be best but would require too many (300) subjects. Instead, four techniques (content-only, temporal locality, causality, and randomizing the results from content-only) were compared against each other using a repeated measures experiment. Twenty-seven non-CS undergraduate test subjects used the search system for a month, and all queries were recorded. In a single test session afterward, seven successful queries were chosen (i.e., the user selected at least one result). For each query the results of the four search algorithms were presented side by side, and the user rated each on a scale of 1–5.

Causal relationship–enhanced search was found to be rated somewhat better (17%) than temporal-enhanced search, temporal was statistically indistinguishable from content-only search, and randomized results were 36% worse. Shah speculated that the reason for the superiority of the causal approach was its improved handling of noise—it added fewer false relationships. Finally, some performance results of the tracing overhead and search overhead were given; they were not sizable but could use some tuning.

Sam closed by expressing the hope that this work would inspire the OS community to consider user studies in their own work. He stated that it really wasn't that painful and that there were many important insights to be gained.

The first question dealt with trace overhead; more efficient methods evidently exist and could be used, with some effort. Another questioner asked about the similarity to Web search and whether this algorithm would apply there or page rank would apply here. Sam pointed out that page rank is based on links that are deliberately created, whereas this system infers links automatically. Terrence Kelly asked why there aren't many proper controlled user studies; the perception is that it's hard and that the IRB is a barrier. Sam stated that it isn't very painful, but that if you can get work published with poor studies, there's little incentive for good ones. Mike Abbot asked what happens if you combine causality and temporal relationship; this is a topic for future work.

- ***Supporting Practical Content-Addressable Caching with CZIP Compression***

KyoungSoo Park and Sunghwan Ihm, Princeton University; Mic Bowman, Intel Research; Vivek S. Pai, Princeton University

KyoungSoo Park presented CZIP, which uses a compression scheme based on content-based naming, where blocks of data are referred to by hashes of their contents. This allows for redundancy elimination, as the same block of data may be incorporated by reference in multiple locations or multiple files. It is used in a number of systems, including LBFS, VBWC, Venti, and others. CZIP defines a format for encoding and decoding files using content-based naming, as well as components to easily add this functionality to existing or new applications.

A number of uses for content-based naming were described: file distribution (e.g., a Linux release, where the contents of the CD ISOs are duplicated in the DVD ISO); virtual machine (VM) images (e.g., migration), where the base OS is identical across machines, and uncacheable Web content, which typically changes slowly and in portions.

CZIP splits a file into chunks, using either a fixed chunk size or Rabin's fingerprint, hashing the chunks, and then representing the file by the sequence of chunk hashes plus the chunks (possibly compressed) themselves. In network applications CZIP can be deployed on the server side, the client side, or both. On the server side, CZIP can either be used to compress files or for in-memory caching only; on the client (or proxy) side a range request can be used to read the header of a CZIP-encoded file, and then only those chunks not already cached need be retrieved.

CZIP compressed the Fedora Core 6 release by a factor of 2; because of duplication between CD and DVD ISOs, bzip2 achieved no compression. On data without duplication (the Wikipedia DB) CZIP gave no compression, whereas bzip2 gave 4x compression. Apache and MySQL server VM images showed high overlap (compressibility) with most parameters, and five VMs used as engineering desktops for three weeks showed very high (96%–99%) overlap. Finally, samples taken every 30 minutes of a number of dynamic Web pages (Google News, CNN, others) showed data savings of 37% to 90% with CZIP. A CZIP-aware Apache server serving the Fedora Core 6 distribution to 100 clients achieved a memory savings of a factor of 2, resulting in higher throughput as the compressed working set fit within physical RAM. A content distribution network (CDN) based on Codeen was shown to decrease origin server bandwidth used to one-quarter that of the non-CZIP version.

A CZIP release at http://codeen.cs.princeton.edu/czip should be available soon.

Terrence Kelly asked why the authors claim that this technique can avoid a round-trip delay compared to duplicate transfer detection (DTD); evidently DTD needs another RTT to check the checksum, whereas CZIP can avoid the delay if it expects and gets redundancy. Jason Flinn asked how useful CZIP is for an individual client, and whether there were any numbers for its effect on client latency. Although the server gets more benefit, clients could still benefit, and experiments to measure client latency haven't been performed yet.

- ***Short Paper: Implementation and Performance Evaluation of Fuzzy File Block Matching***

Bo Han and Pete Keleher, University of Maryland, College Park

Bo Han presented an evaluation of fuzzy file block matching, an extension to content-addressable storage for efficiently representing small updates to files. Instead of identifying each block by a single hash, and only substituting exact matches, a hash vector that can be used to measure block similarity is produced for each block; error-correcting codes may then be used to "correct" a similar block to create the desired block. This extends work by Tolia et al. (USENIX '03) by providing a replicatable implementation and performance evaluation.

Files are chunked via Rabin's fingerprint, and then each block is described by a "shingleprint." These are computed by taking a sliding m-byte window within the block, starting at each byte offset and hashing that window, and then subsampling the resulting set of hashes by taking the S hashes with the smallest numerical values. The fuzzy matching algorithm declares two blocks to be similar if their prints share T out of S values. An ECC code for each block is then generated by using a 255/223 Reed-Solomon code on small parts of the block. Finally, a possible architecture for using fuzzy matching to maintain and update a cache was described.

Experiments were performed using the GNU Emacs source tree versions 20.7 through 21.4, counting how many files in version *N*+1 could be recovered from version *N* plus ECC information for version *N*+1. These experiments were performed for a number of different parameters, including the sliding window size and the ECC organization.

Related work includes Venti, which uses hashes as block IDs, REBL, which uses super-fingerprints, TAPER, which uses Bloom filters, and LBFS. Bo concluded by saying that the main advantage of fuzzy matching is the possible savings of network bandwidth and that more experiments with expanded data sets, ECC codes, and parameter combinations were needed, as well as integration of fuzzy matching into an existing distributed file system.

Terrence Kelly asked whether any evaluation of the effectiveness of this proposal in the wide area would be difficult because of the need for traces of both requests and replies. Bo replied that they had only looked at the local version; future work might examine this, and in that case it might well be an issue. To a question about hash collisions and security Bo replied that these may need consideration in the future as well.

INVITED TALK

- ***Live Malware Attack!***

Paul Ducklin, Sophos

Summarized by Jan Stoess (stoess@ira.uka.de)

Paul Ducklin presented a demonstration of how typical malware exploits a computer system, and how a malware analyst can find out the nature and behavior of such malware. Since malware usually relies on Internet access, an analyst must provide a "simulated Internet" environment that causes the exploit to install and activate itself but also allows monitoring and inspection of how the exploit proceeds.

Paul presented such a "deductive and analytical" inspection environment, which consisted of two virtual computers running within the QEMU simulator on his host laptop. The first virtual computer contained a Windows installation acting as the client, and the second hosted a Linux instance establishing the "simulated Internet." The simulation consisted of a set of tools providing standard services such as DNS, HTTP, and Mail in a way that client requests could be monitored and transparently rerouted to services or files in the control of the malware analyst.

Paul then installed a sample malware program in the Windows client. The program exploits a bug in the library code processing Windows Meta Files (WMF). The malware attack occurs via a WMF file embedded in a Web page. Since the library routine does not check for overflows during retrieval, it also copies the malware program on the stack; it then overwrites the stack return address to activate the malware code.

Paul then proceeded to inspect the code within the file. It turned out to be scrambled, and no additional knowledge could be inferred without installing it into memory. Paul therefore downloaded the malware again, but this time he ran the browser within a debugger. In addition, he modified the malware's source code to trap into the debugger after activation. Paul could then single-step the code, to observe that the next steps would be to download a Trojan horse rootkit via a SOCKS proxy. Paul executed that rootkit together with a file system call monitor; he could thereby determine whether the rootkit had installed itself into several new files but had also hidden itself from inspection in the file system.

Paul ended his enjoyable presentation by demonstrating how the rootkit actually stashed itself. For that purpose, Paul attached WinDbg, a local kernel debugger, to the client Windows installation. The malicious driver had changed the Windows system call table to transparently rewrite file system access calls in such a way that they would ignore the newly generated files containing the rootkit code.

SYSTEM SECURITY

Summarized by Francis David (fdavid@uiuc.edu)

- ***From Trusted to Secure: Building and Executing Applications That Enforce System Security***

Boniface Hicks, Sandra Rueda, Trent Jaeger, and Patrick McDaniel, Pennsylvania State University

Boniface Hicks started by pointing out that mandatory access control (MAC) works well for naturally separated processes. Several operating systems use MAC to restrict the flow of information between processes based on security levels. Unfortunately, there are several applications that defy classification at a security level and are marked as "trusted" and are allowed complete access to the system. An example of such an application is a log rotation server that needs to rotate logs for multiple applications at different security levels. The goal of this research is to allow MAC policies to be enforced within applications as well, in order to prevent their circumvention by compromising a trusted application.

To enforce system security policies within applications, Boniface proposes the use of security-typed languages. An operating system service called SIESTA has been designed and built that can enforce SELinux security labels at compile time within applications written in the Jif security-typed language. The security labels used by the operating system are tracked within the application and the language ensures that information flows are in compliance with the MAC policies. Declassification or downgrading the security level of an information flow is also allowed if it is specified in the policy.

A secure logrotate service and a secure email client were written to demonstrate this concept. Experiments show that the performance of these secure applications suffers some deterioration when compared to native applications. This overhead is close to 100% for logrotate. Boniface mentioned that this may be because the Jif code was not optimized. In his response to a question, he noted that the system does not avoid covert channels.

- ***From STEM to SEAD: Speculative Execution for Automated Defense***

Michael E. Locasto, Angelos Stavrou, Gabriela F. Cretu, and Angelos D. Keromytis, Columbia University

Michael Locasto said that self-healing systems are needed because typical computer defense systems crash the process that is being protected during an attack. Self-healing systems provide a less drastic approach to recovering from an attack. This work extends and addresses some shortcomings of their previous system for self-healing, which was called STEM. The new system provides self-healing capabilities for unmodified binaries without the need for source code. In STEM, every function is treated as a transaction that is speculatively executed. Error codes are returned when a problem is detected. The SEAD system improves on this design by utilizing binary rewriting to avoid source code modification and recompilation. It also supports repair policies to customize an application's response to an attack. Virtual proxies are used to ameliorate the output commit problem, and process behavior profiles can be created on aspects of data and control flow.

Repair policies can be used to perform better error virtualization and avoid returning incorrect return values. They also support memory rollback for aborted transactions and forced modifications to memory. Another advantage of using repair policies instead of generating and deploying new code is that a policy can easily be switched off if it turns out to be incorrect.

Performance evaluations show that although the system does not have a human-discernable impact when applied to regular applications, there is a rather significant impact on an application's startup time. As part of future work, Michael is looking at ways to automatically generate repair policies.

Several members of the audience pointed out that software vulnerabilities will still exist after the system self-heals and suggested that it might be better just to fix the code than to fix bad code with policies. Yet another limitation that was pointed out was that it is not easy to decide upon the meaning of return values when there is no access to source code.

- ***Dynamic Spyware Analysis***

  Manuel Egele, Christopher Kruegel, and Engin Kirda, Secure Systems Lab, Technical University Vienna; Heng Yin, Carnegie Mellon University and College of William and Mary; Dawn Song, Carnegie Mellon University

Manuel Egele began his presentation by noting that spyware is a growing threat to Internet users. Browser Helper Objects (BHOs) are widely used to implement spyware and are the focus of this work. Detection of such spyware using signature-based techniques misses newer spyware, and behavior-based techniques are required to detect them. The system described in the presentation tracks the flow of sensitive data through the system and observes the behavior of BHOs. For example, URLs that are typed in are considered sensitive information; the system checks for BHOs that misuse this information. The system provides comprehensive reports about file, network, IPC, and OS actions.

QEMU was modified to enable the tracking of data and control dependencies. The control dependencies tracking approach makes use of a control flow graph generated from disassembled instructions. A browser session of a user is prerecorded and this captured session is replayed for analysis of spyware behavior. Experiments show that the system is effective at detecting and presenting an analysis of several spyware samples. It was also able to detect a spyware sample that was not detected by several commercial products. Manuel noted that running the analysis using the QEMU emulator caused a factor-of-10 slowdown in execution.

INVITED TALK

- ***LiveJournal's Backend Technologies***

  Brad Fitzpatrick, LiveJournal

  Summarized by Peter J. Desnoyers (pjd@cs.umass.edu)

Brad Fitzpatrick described the process of evolving the LiveJournal implementation from the college hobby project for college and high school friends in 1999 to the system it is today, with over 10 million accounts. Slides for this talk (or at least equivalent talks) are available at http://www.danga.com/words.

LiveJournal combines blogging, forums, social networking, and aggregation. In its current form it includes the following open-source components, all of which evolved as the system scaled: memcached, mogilefs, perlbal, gearman, theschwarz, djabberd, and openid.

Brad started by discussing scaling, and in particular he said that the absolute speed of a solution isn't as important as whether it scales linearly—does it run 2x or 10x as fast when you use 2x or 10x as many servers? If not, you'll reach a point where you have to restructure the system to grow, which they did several times.

LiveJournal started out on a single machine with Apache and MySQL, and it worked fine until it got slow. The Apache and MySQL servers were split onto two machines, which created two points of failure and soon became CPU-bound on the Apache machine. This was replaced by three servers with load balancing—a number of solutions were used, but none was completely satisfactory.

Then it became I/O-bound on the MySQL server, which was split in two with MySQL replication. This scaled to twelve mod_perl (application) machines and a MySQL master with six slaves. Then database writes became the bottleneck: Each read was handled on one MySQL machine, but writes were broadcast to the cluster, so all the MySQL machines were spending all their time writing.

This problem was solved by partitioning the database. Users map to one of a number of high-availability MySQL clusters, with a single global master and slave for nonuser metadata. Each cluster is a high-availability pair using the DRBD network block driver for shared storage, plus slaves to add more read capacity.

In the rest of the talk Brad described the components they had developed.

Memcached is a single cache for everything, running wherever there is spare memory. It presents a simple dictionary data type with a network API, and the server for a particular data item is determined by hashing. The main problem is that you can't add or remove servers at runtime without rehashing everything, but this has been solved recently by using consistent hashing.

Perlbal is a "fast, smart, manageable HTTP Web server/reverse proxy/load balancer." It has two key features: internal redirects, allowing a backend to hand-work (e.g., serving a large file) back to perlbal without the user seeing a redirect, and verification of backend connections.

Mogilefs is a simple read-mostly replicated file system, which, like memcached, can be deployed wherever there are resources.

Gearman is a distributed job/function call system. Workers register functions they implement, and callers invoke with function and arguments encoded as simple strings. Among the uses mentioned were database connection pooling and keeping large libraries (e.g., ImageMagick) out of the mod_perl processes that needed to invoke them.

Theschwartz is sort of like gearman, but instead of being lightweight and unreliable (i.e., the caller has to wait and retry if necessary), theschwartz is reliable and can be used to schedule something and forget it, such as sending email after updating a page.

Djabberd is needed because the other XMPP servers weren't flexible enough to integrate closely with the rest of LiveJournal (e.g., automatically use the existing LiveJournal user picture as an avatar). It's lightweight and just about every function can be hooked.

Ted Ts'o asked several operations questions: Brad guesses they have 150–200 machines, they still run memcached anywhere there's spare memory, and they put disks into netbooting Web nodes for mogilefs, and both practices drive operations batty (a recurring theme). They only have a single data center, but they are expanding into another in Oakland. In response to a question from Simson Garfinkel about their multiday outage in 2005, Brad pointed out that geographic redundancy is much harder and more expensive than redundancy within the same facility.

Although the operations people are scared of it, they use virtualization (Xen), especially to isolate applications that might "explode" and use too much memory (ImageMagick again). Configuration was originally generated from a single YAML file; it is being migrated to cfengine. Failures were discussed; a surprising number were due to other customers at the same facility pressing the Big Red Button. This brought up the issue that it's hard to get a storage stack to sync properly; for example, often even with battery-backed RAID cards the disks are running in write-back mode, and data the DB thinks is committed will be lost if power goes out.

CLOSE TO THE HARDWARE

Summarized by Andrew Baumann (andrewb@cse.unsw.edu.au)

■ ***Evaluating Block-level Optimization Through the IO Path***

Alma Riska, Seagate Research; James Larkby-Lahet, University of Pittsburgh; Erik Riedel, Seagate Research

Optimizations in the disk IO path are traditionally thought to be more effective at high layers (such as the file system) where more semantic information about requests is available. However, with advances in modern disk hardware allowing implementation of complex optimizations such as request reordering, this work presented by Alma Riska evaluates the impact of performing optimizations at the disk level. The approach used is based on measurements of the Postmark benchmark and kernel compiles on Linux, with variations in the file system, IO scheduler, and disk drive (with each of the drives used supporting request reordering).

Results show that higher in the IO path, at the file system and IO scheduler, the focus should be on optimizing the amount of disk traffic. The best-performing file systems and IO scheduler algorithms were those that achieved better rates in IO workloads. Lower in the IO path, the focus is on reordering and coalescing requests, which is most efficiently performed at the disk level.

One particularly interesting result was that under write-back caching, which is generally assumed to perform better than write-through caching, increasing the queue depth yields inconsistent performance. Under high load the disk queue fills up, causing more requests to block at the device driver, which is unaware of the disk's queue. Write-through caching with queuing and reordering performs as well as write-back without queuing, and it doesn't suffer from reduced reliability in the case of power failures.

One audience member observed that an advantage of write-back caches was the ability to merge repeated writes; however, Alma noted that previous work at the 2006 conference had shown that repeated writes were extremely rare. She also acknowledged that the performance of write-through and write-back caching was only equivalent in regard to throughput and that the response time for write-through caches would be higher.

■ ***DiskSeen: Exploiting Disk Layout and Access History to Enhance I/O Prefetch***

Xiaoning Ding, Ohio State University; Song Jiang, Wayne State University; Feng Chen, Ohio State University; Kei Davis, Los Alamos National Laboratory; Xiaodong Zhang, Ohio State University

Xioning Ding presented this work on DiskSeen, a prefetching scheme that uses knowledge of the disk layout and access history information to improve on the performance of traditional file-level prefetching, which can incur nonsequential accesses and thus result in excessive seeking.

DiskSeen operates below the file-system level, maintaining a disk block table storing the access history of logical block numbers, which it uses to detect sequences for prefetching. Two types of prefetching are supported: sequence-based prefetching, which occurs when eight or more contiguous blocks are accessed, and history-aware prefetching, which uses the disk block table to detect and prefetch historic access trails that may not be contiguous.

A performance evaluation of DiskSeen in Linux shows that it reduces the time taken for repeated runs of prefetching-friendly applications by 20%–50%. The time for the first run of each application is also improved, but not as much, owing to the lack of history trails in the disk block table. The performance improvement is greatest for applications, such as CVS, that access multiple areas of the disk and thus have the greatest seek overhead. One TPC benchmark degraded by 10% with DiskSeen; however, the authors plan to fix this problem with dynamic adjustment of the timestamp threshold.

- ***Short Paper: A Memory Soft Error Measurement on Production Systems***

  Xin Li, Kai Shen, and Michael C. Huang, University of Rochester; Lingkun Chu, Ask.com

This study, presented by Xin Li, looked at the frequency of soft errors, which are transient hardware errors in memory caused by environmental factors. Previous studies have suggested error rates in the range of 200–5000 failures in time (FIT) per megabit; however, there are no results for modern hardware in a production environment.

This study used several data sources: a server farm at Ask.com, a number of desktops at the University of Rochester, and nodes on PlanetLab. Results for the Ask.com servers were collected from ECC memory error statistics; results from the other machines were collected by a user-level application that attempted to detect memory errors by writing a pattern in memory and checking it for errors. In total over 300 machines were used; however, the 70 PlanetLab nodes were only able to allocate 1.5 MB of free memory in which to look for errors.

The only transient errors found were two in the Ask.com server farm. This corresponds to an error rate orders of magnitude lower than previously reported. The study also unintentionally found some permanent errors (nine errors in the 212 Ask.com machines). The authors conclude that the actual transient error rate is much lower than previously reported, and they speculate that this is due to changes in hardware layout and a reduction in chip size. Future work includes further data collection, identifying error modes that can escape hardware protection, and studying how real software systems will be affected by such errors.

INVITED TALK

- ***MapReduce and Other Building Blocks for Large-Scale Distributed Systems at Google***

  Jeffrey Dean, Google

  Summarized by Jan Stoess (stoess@ira.uja.de)

Jeffrey Dean gave a presentation on how Google manages access to the billions of documents available on the Web and via other Google-provided services such as email, personal files, its closed database system, broadcast media, and print services.

Jeffrey explained that his company faces an ever-increasing computational need, which is driven mainly by three factors: (1) more queries, from more people using Google and using them more frequently; (2) more data from the growing Web community and from new products; (3) the need for better search results (finding the right information and finding it faster). Jeffrey then presented three software projects Google has developed to address the increasing computational requirements: GFS, MapReduce, and Bigtable. Following the hardware design philosophy of Google, the software systems run on a very large number of commodity machines, which offer a good price/performance ratio.

The large number of nodes and the huge amount of data lead to unique requirements for the file system. Google has therefore developed its own distributed file system, named GFS. Since Google has control over all applications, libraries, and the operating system running on its machines, it can optimize access to the file system on all levels. A GFS setup consists of a master server holding the metadata and multiple chunk servers holding the actual data. Each file is broken into chunks of 64 MB, each of which is stored redundantly on multiple chunk servers. The chunk servers use a standard Linux file system. Currently, Google stores more than five petabytes of data using GFS.

For processing the data, Google has developed a special programming environment called MapReduce. A MapReduce program consists of a map function and a reduce function. The map function extracts relevant information from each input record and stores it as key/value pairs. The reduce function then aggregates the intermediate values, for instance by means of a hash function. As a simple example, Jeffrey explained how the word frequency of an input text can be determined with MapReduce. In this case the map function splits the input at spaces and has the value "1" for each word. The reduce function sums all values that have the same key. The MapReduce environment is implemented as a library that deals with most of the technical aspects such as the highly parallel and distributed execution of the algorithm, or the reading and writing of data. It thus drastically reduces the programming effort and lets MapReduce users focus on their real goals. The MapReduce implementation is optimized to be fast, robust, and scalable; it has been used for many Google applications such as Google Ads, Froogle, Google Earth, and Google News. As of June 2006, Google had developed more than 6000 MapReduce programs.

Finally, commercial database management systems do not perform well with the amount of data Google has to manage. Commercial systems also have high licensing costs. Furthermore, the special nature of the stored data and the way of processing it allow for several simplifications and optimizations compared to full-featured database management systems. Google has therefore implemented its own database management system, called Bigtable. The basic data model of Bigtable is a distributed multidimensional sparse map. Each cell is identified by a (row, column, timestamp)-triple. A Bigtable for Web pages, for example, can hold different properties (column) of a Web page located at a specific URL (row), using multiple versions (timestamp). Bigtable is used for several Google products, such as its Web search and Google Earth.

For more detailed information, Jeffrey referred to http://labs.google.com/papers.html, a page featuring one paper for each of the three presented technologies.

NETWORKED SYSTEMS

Summarized by Xiaoning Ding (dingxn@cse.ohio-state.edu)

- ***Addressing Email Loss with SureMail: Measurement, Design, and Evaluation***

Sharad Agarwal and Venkata N. Padmanabhan, Microsoft Research; Dilip A. Joseph, University of California, Berkeley

The paper addresses the silent email loss problem. In the paper, a silent email loss is defined as the case in which an email is never received by the intended email recipient but neither the sender nor the intended recipient is notified of the loss. Sharad Agarwal first presented the results of their measurements on email losses, and then presented their design of SureMail. SureMail augments the existing SMTP-based email system by notifying the intended email recipients about email losses.

Sharad Agarwal and his team performed an experiment to understand how often legitimate user emails were lost. In the experiment, many email accounts were used to send and to receive emails over several months. Then the sent emails and the received emails were matched to check email losses. The measurements show a silent email loss rate ranging from 0.71% to 1.02% and a total email loss rate ranging from 1.82% to 3.36%. They also compared the email loss rates for two normal msn.com accounts and two msn.com accounts with content filters disabled and found that the accounts with disabled content filters have much lower loss rates. This indicates that the majority of losses were from content filters.

Because the measurements show that the existing SMTP-based email system works most of the time, SureMail is designed, not to replace the existing system, but to augment it with a separate notification mechanism. A notification is a short, fixed-format fingerprint of an email. When an email is sent, the notification is also delivered via an in-band channel or an out-of-band channel. The in-band channel uses email headers, and the out-of-band channel uses separate services such as DHT, Amazon S3, or dedicated notification servers. To prevent spoofing by spammers, SureMail uses a reply-based shared-secret scheme. The scheme sets up a shared secret based on an email and its reply between two correspondents, and it uses the shared secret to authenticate the notifications.

The evaluation on the out-of-band channel shows that 99.9976% of notifications can be delivered successfully at a very low incremental cost.

- ***Wresting Control from BGP: Scalable Fine-Grained Route Control***

Patrick Verkaik, University of California, San Diego; Dan Pei, Tom Scholl, and Aman Shaikh, AT&T Labs—Research; Alex C. Snoeren, University of California, San Diego; Jacobus E. van der Merwe, AT&T Labs—Research

Patrick Verkaik presented the design and implementation of IRSCP (Intelligent Route Service Control Point), which is an architecture enabling flexible route control for inter-domain traffic without changing existing ISP infrastructure. IRSCP is the follow-up work on RCP.

In the IRSCP architecture, a route control application uses external information, such as network condition, to guide the route selection process in IRSCP. IRSCP communicates the selected routes to the routers in the ISP network and in the neighboring ISP network. The route control application works at relatively slow rate. To enable IRSCP to failover instantly in case a route for egress link fails, the route control application provides IRSCP with an explicit ranking of egress links for each ISP router. To prevent forwarding anomalies caused by inconsistent rankings, such as deflection and looping, IRSCP enforces two simple consistency constraints on the rankings. In a large ISP, IRSCP communicates with many thousands of routers, and it is responsible for route decision for each ISP router. Therefore, IRSCP must be robust and scalable. The paper addresses the problems by partitioning and distributing the IRSCP functionality across multiple IRSCP servers.

Patrick Verkaik and his team evaluated IRSCP by connecting IRSCP to an emulated ISP. The results show that IRSCP is capable of managing the routing load of a large ISP. In response to a question about whether converging could be a problem, Patrick Verkaik replied that IRSCP converges quickly.

■ ***A Comparison of Structured and Unstructured P2P Approaches to Heterogeneous Random Peer Selection***

Vivek Vishnumurthy and Paul Francis, Cornell University

Vivek Vishnumurthy talked about the heterogeneous peer selection problem in structured and unstructured P2P networks. Heterogeneous peer selection means that peers with higher capacities should be selected proportionately more often. To support heterogeneous peer selection, Vivek and Paul implemented Swaplinks (published at Infocom 2006) for unstructured P2P networks and adapted Bamboo DHT, using their extensions to the Karger/Ruhl load-balancing algorithm for structured P2P networks. Then they compared the performance of Swaplinks and the adapted Bamboo DHT (called KRB) in making heterogeneous selections based on capacities.

The basic idea of Swaplinks is to build a random graph, in which each node tries to keep its degree proportional to its specified capacity. Thus the unbiased fixed-length random walks on the random graph result in the desired selection probability. The basic idea of KRB is to adjust peers' ID spaces dynamically based on their loads and their capacities, so that all the peers have close relative loads. The relative load of a peer in KRB is its load divided by its capacity.

The authors tested Swaplinks by emulating a 1000-node P2P network on 20 CPUs and evaluated KRB by simulating a same-scale P2P network. Three conclusions were drawn from the comparison: (1) Swaplinks makes more accurate selections than KRB does; (2) KRB's performance approaches Swaplinks only under low churn and moderate capacity distribution; (3) KRB is sensitive to parameters, and it is harder to set optimal values for the parameters in KRB than in Swaplinks.

INVITED TALK

■ ***Perfect Data in an Imperfect World***

Daniel V. Klein, Consultant

You can find a summary of Dan's talk in the April 2007 issue of *;login:*, in the summaries of LISA '06.

KERNELS

Summarized by Rik Farrow (rik@usenix.org)

■ ***Transparent Checkpoint-Restart of Multiple Processes on Commodity Operating Systems***

Oren Laadan and Jason Nieh, Columbia University

Oren Laadan explained that modern applications consist of multiple processes, so we need a method for capturing global state. This mechanism should be transparent for both applications and sysadmins to use, and it should also not require kernel modifications. Current approaches use modified libraries (an incomplete solution), modified kernels (which is invasive and difficult to maintain), and the use of hypervisors (which implies adding an OS layer and more overheard).

Their approach uses a loadable kernel module that virtualizes just the set of processes to be checkpointed, called a POD, or PrOcess Domain. A POD has a private virtual namespace and is decoupled from the OS. Checkpointing uses auxiliary processes with COW (Copy On Write) and buffers that hold data until it can be committed. Checkpointed processes can have their data filtered to compress it, transform data for another OS version, or adjust data structures. Quiescing a process can be done with SIGSTOP, forcing a known state with minimal stack synchronization. A Process Forest is the set of dependent processes, related either via a parent process or through shared resources all within the same process group.

Oren showed sample output of the DumpForest algorithm, which includes information on dead processes as an artifact of the design. He compared the performance of Checkpoint to OpenVZ and XEN, showing that checkpoint and restart times were 3 to 55 times faster than OpenVZ and 5 to 1100 times faster than Xen. Checkpoint times, at 100 ms, were fast enough not to be noticed by a human being.

Warner from Google said that he had worked with checkpoint in the early 1990s and wondered about network connections in flight, files, and the fact that some processes expect to see the same PIDs after restoration. Oren answered that PIDs are virtualized, isolated via POD, so they don't change. Filesystem issues are addressed using filesystem snapshotting technology. Network connections that are inside the POD are easy to handle. For those outside the POD, we do have a way to move connections if the lag time is short. We can even restart connections on the other side transparently. Someone else asked why the performance is so different than that with the OpenVZ approach. Oren said they were surprised and puzzled too. In restart, OpenVZ was always 1 second longer in OpenVZ. For checkpoint time, they think that teardown and freezing takes longer. They can freeze a process in 1–3 ms. Phil Pennock of Google asked what security analysis had been done and the implications for SUID programs. Oren replied that you have to trust the image that you are restarting.

■ ***Reboots Are for Hardware: Challenges and Solutions to Updating an Operating System on the Fly***

Andrew Baumann, University of New South Wales and National ICT Australia; Jonathan Appavoo, Robert W. Wisniewski, Dilma Da Silva, and Orran Krieger, IBM T.J. Watson Research Center; Gernot Heiser, University of New South Wales and National ICT Australia

Andrew Baumann presented this paper about dynamic updates to the K42 operating system. Previous work exists for applications but is unsuitable for an OS because of low-level languages and concurrency issues in the kernel. They enabled dynamic update by using modularity in the kernel, in contrast to prior work (DynAMOS and LUCOS) that applies patches by rewriting code on the fly, or AutoPOD, which uses virtualization. This work fits somewhere in the middle and uses modules to focus on maintenance changes.

K42 is a scalable research OS that supports Linux API/ABI, is object-oriented, and has each resource managed by a set of object instances. All objects go through an object translation table (pointers) that allows substitution of objects on the fly. A dynamic update is just a series of hot swaps, but you need to replace every module affected. The previous work on K42 would allow some updating via hot-swapping, but not those that include changes to interfaces. A total of 58% of changes affect interfaces in K42. They looked at stable kernel releases of Linux kernels and saw that more than half of the changes affected interfaces as well.

To support dynamic updates, you write an adaptor that can rewrite calls via the old interface to support new function parameters. In testing, the adaptor has 220 cycles of overhead. First, you update the provider object with an adaptor, then update the clients of that object so that they use the new interface, then remove the adaptor. This only works for backward-compatible changes and accounts for about 80% of the changes to the K42 kernel over its history.

Someone asked, What about the remaining 20% of the changes? Andrew answered that outside of module code, such as low-level exception handlers, you can't use this technique. But sometimes you can move the change to a module instead of outside one. Applying a patch produced a drop of 10% while running ReAIM throughput benchmark, with 170 files open. In summary, there is negligible performance impact and 79% of maintenance changes can be updated.

In response to Francis David's question of when it is safe to apply the patch, Andrew explained that the patch is applied as a series of hot-swap operations and must achieve quiescence of the object first, and that makes it safe. Terrence Kelly of HP Labs asked whether it was safe to apply another patch when another lazy update is in process. If you applied another patch, it would mark all the objects as needing updates again, but not break anything.

- ***Short Paper: Exploring Recovery from Operating System Lockups***

  Francis M. David, Jeffrey C. Carlyle, and Roy H. Campbell, University of Illinois at Urbana-Champaign

Francis David started by mentioning that Linux has a lockup detector that works via a timer interrupt that checks a timestamp for continual updates via a low-priority kernel thread. But this mechanism fails if a lockup occurs when interrupts have been disabled. Watchdog timers present an external alternative, as they can generate a nonmaskable interrupt (NMI) if they time out, forcing the system to reboot. However, such reboots mean loss of state, even though the contents of RAM may still be correct. Francis pointed out that they did their work on ARM CPUs that do not include support for NMI.

In their approach, they replace the interrupt handler for the NMI with code that resets the processor, enables the MMU, reinitializes the interrupt controller and interrupts, and then modifies locked-up threads. In Linux, they simply kill off the offending threads, but in Choices, an object-oriented OS, they patch in a call to an exception handler and restart scheduling. If these threads hold locks, the locks will be released when the threads die. Choices is fully preemptable, and the patched-in recovery routine pretends to be a thread that is locked up and calls die(), or raises a C++ exception so programmers can decide what to do at this point. For this scheme to work, there must be valid context via the stack frame of the thread running before the NMI.

Francis summed up by saying that the best lockup detection uses both hardware and software detection: hardware when software cannot work, and software for preemptible kernel, where hardware cannot detect failure. Their approach improved recovery in Linux up to 9%, particularly with preemptive kernels.

Someone from the University of Rochester asked whether they explored using NMI at all. Francis said that they did, and their code shows examples of this. They wrote the code to actually recover from an NMI as well. Ben Leslie of Open Kernel Labs pointed out that sometimes the state in the OS has been corrupted. You get one watchdog reset, and you keep on doing this. Do you need a meta-watchdog? Francis's team didn't explore corruption issues, or whether the same issue happens twice. Daniel Peek of the University of Michigan asked how much of the kernel malfunction problem this will solve, for example, kernel panics. If you get a "blue screen," the OS has detected the problem. The authors are addressing the lockup problem when the OS can't recognize the error. Someone else asked how often that happens. In the Linux kernel, the majority of possible bugs were lockup bugs (30% or more bugs could cause an infinite loop or other problem in Stanford static code analysis).

INVITED TALK

- ***Human Computation***

  Luis von Ahn, Carnegie Mellon University

  Summarized by Minas Gjoka (mgjoka@uci.edu)

Luis von Ahn started by defining the term CAPTCHA (completely automated public Turing test to tell computers and humans apart) as a program that can distinguish humans from computers.

Luis gave a basic example of the usefulness of CAPTCHAs based on a true story in 1999. Slashdot released an online poll letting its users select the best CS grad school from a list of six universities. The only safety measure to prevent manipulation of the poll results was to allow one vote per unique IP. However, in a matter of hours Carnegie Mellon and MIT students managed to write programs that would cast thousands of votes into the system. This example demonstrated the need for a mechanism that will only allow humans to participate.

Other applications of CAPTCHAs include free email services, worms, data collection, prevention of comment spam in blogs, and dictionary attacks. For example, in free email services, spammers are prevented from signing up millions of accounts automatically. One workaround for spammers is the use of sweatshops, which hire people in countries with very low wages to solve CAPTCHAs for them. That incurs a minimum penalty per account creation for the spammers. Another workaround is to redirect CAPTCHAs from email service companies to the spammer's own Web sites, which provide services that attract many people (e.g., porn sites). In another example, email addresses can be protected from Web crawlers by using CAPTCHAs.

One of the main aspects of the presentation is the usage of CAPTCHAs to perform human computations. A measure of the amount of human effort produced daily: it is reported that around 60 million CAPTCHAs are solved every day, with each CAPTCHA taking 10 seconds of human time. Three programs are presented to make good use of these wasted "human cycles."

CAPTCHAs can be used to help in the digitization of old books. Every scanned image of a word not recognizable by OCR is used as a CAPTCHA. To confirm the correctness of the human input, every time a scanned image of a word is fed into the system it is combined with a known word. If the answer for the known word is correct then a correct answer is assumed for the unknown word as well.

A very useful application of CAPTCHAs is to accurately label images with words. Luis has developed an enjoyable two-player online game, ESP, which was designed in such a way that playing the game results in labeling images correctly, quickly, and for free. At the beginning of the game the user is paired with another random player and the same image is shown to the two players. The goal of the game is to guess descriptions of the image that are identical for both players, excluding taboos. The two players cannot communicate in any way with each other and anticheating techniques are provisioned for potential collaborators. Luis notes that this game alone could be used to label all Google images within a few weeks. In fact Google offers a similar "Google Image Labeler" service. Those who liked using the game listed various reasons (e.g., it offers a special connection with one's partner, it helps one learn English, and it gives one a sense of achievement).

PeekaBoom is another entertaining two-player online game that takes as input labeled images and finds the objects being labeled. The first player, called Boom, receives an image and a tag assigned to the image. The second player, called Peek, has an empty screen. The goal of the game is to get Peek to guess the tag assigned to the image. Boom can only reveal part of the picture. In addition to that, Boom can give hints about what the tag is (e.g., noun, verb, text in the image).

By combining the region selected for a given object from different pairs of players in PeekaBoom it is possible to get the whole outline of the object in 50% of the cases. This allows the results to be highlighted with boxes inside the search engine. Another advantage of this segmentation is that the resulting training set could be used to advance computer vision research.

In conclusion, the speaker presented a paradigm for dealing with open problems in artificial intelligence. These can be turned into either a test to differentiate between humans and computers or a simple game that people can play online.

In response to a question Luis noted that, after 20 hours of playing, the gender of a player can be guessed with 98% accuracy, and the age with 85% accuracy.

SHORT PAPERS

Summarized by Andrew Baumann (andrewb@cse.unsw.edu.au)

■ ***Short Paper: Supporting Multiple OSes with OS Switching***

Jun Sun, Dong Zhou, and Steve Longerbeam, DoCoMo USA Labs

Dong Zhou presented this work on a mechanism to switch between operating systems on shared hardware. Each OS is assigned a unique range of physical memory and has exclusive access to the hardware when it runs. The switch is performed when a switch request signal is received; this can be generated by user action or by events such as timer expiration or incoming call. The OS in the background is essentially suspended, it cannot receive interrupts, and there is no regular time-slicing. The switch operation consists of putting the hardware into a consistent state and then passing control to the other OS.

OS switching is usually implemented as a modification of the existing suspend-and-resume support, so relatively little code is changed in the operating system. Furthermore, because each OS runs with direct access to hardware, there is no slowdown. Limitations of the approach include a lack of concurrency and no security between the OS instances, although the latter could be addressed with hardware support such as ARM TrustZone.

A prototype has been implemented on an ARM9 device, and a video was shown of the device switching between Linux and Windows CE in response to a special button press. Around 100 lines of code were changed in either OS, and all within the board support packages; most of the modified code was in the bootloader. Switching from Linux to Windows CE takes half a second; switching to Linux takes a second longer, mainly because more devices were enabled than under Windows CE. The speed of switching could be improved by not suspending and resuming all devices; in the extreme case, a hot switch could take less than 1 ms.

- ***Short Paper: Cool Job Allocation: Measuring the Power Savings of Placing Jobs at Cooling-Efficient Locations in the Data Center***

  Cullen Bash and George Forman, Hewlett-Packard Labs

The overall goal of this work, which was presented by both authors, is to reduce the energy used for cooling a data center. This could offer significant cost-savings, because the power used for cooling doesn't scale linearly with the power used to run the equipment (i.e., a rack using twice the power may require much more than twice the power to cool).

Within a data center, some servers can be more efficiently cooled than others because of the varying recirculation of hot air, so the overall cooling workload can be reduced by moving long-running jobs to those servers that are more efficient to cool and shutting down other servers when they are idle. The approach of this work is to modify a scheduler for batch jobs so that the longest-running jobs are placed on the most cooling-efficient servers.

In this study, part of a data center was physically partitioned, and the power consumed by both hosts and air-conditioning units was monitored. The results showed that controlling job placement alone helps, but the greatest power savings come from combining job placement with shutdown of idle servers. This reduced power consumption by 33% and could save $1 million per year for HP data centers. Future work involves incorporating these techniques into adaptive enterprise software.

- ***Short Paper: Passwords for Everyone: Secure Mnemonic-based Accessible Authentication***

  Umut Topkara, Mercan Topkara, and Mikhail J. Atallah, Purdue University

Umut Topkara presented this work, which aims to develop a secure authentication mechanism in input-constrained environments, such as for disabled users. The assumed input device is a binary switch; in this scenario it is hard for humans to remember long bit strings, and it should be possible to initialize passwords with the same input device. Furthermore, the technique should be secure against dictionary, replay, shoulder-surfing, and phishing attacks.

These problems are solved by the PassWit system, which also maps well to traditional plain-text passwords, and thus is compatible with conventional password systems and input devices. At password initialization time, the user is given a random mnemonic sentence selected from a number of word tables. At authentication time, the user is asked a series of yes/no questions, based on the format "Does your mnemonic contain one of these words?"

To avoid record/reply attacks, different questions are asked each time, and to avoid inferring the mnemonic from the questions that are asked, the questions must be determined at the beginning using combinatorial group testing. To protect against spyware, images or CAPTCHA techniques can be used, and the system inherently protects against phishing, because the mnemonic itself is never entered.

- ***Short Paper: Virtually Shared Displays and User Input Devices***

  Grant Wallace and Kai Li, Princeton University

The final paper of the conference, presented by Grant Wallace, covered work on enabling collaboration on shared displays with multiple input devices, for example, the Princeton Plasma Physics Lab control room where a large shared screen is used to allow multiple users to collaborate. Traditional OSes and windowing systems are not appropriate, because they assume the general model of one user at one display with one set of input devices. The goal for this new system is to allow multiple user workstations to connect with each other and the shared display and to allow the users to seamlessly move cursors and windows between the workstation and the shared display.

Traditional collaboration systems (such as X and VNC) are platform-specific, initialization-constrained, support only one-to-many sharing, or share only at the granularity of an entire desktop. To address these limitations the Fusion collaboration system was developed. It uses a modified VNC server to allow sharing at the granularity of windows rather than the desktop, a modified VNC viewer to simultaneously display windows from multiple users, a modified window manager that supports multiple cursors by time-slicing cursor activity to the system cursor, and a modified X2X utility that captures input from multiple users.

The system has been deployed in two locations and has received very positive user feedback. It enables users to share and compare windows while providing better performance and privacy than the desktop sharing of normal VNC. Further details and source code are available at http://shared-app-vnc.sourceforge.net and http://multicursor-wm.sourceforge.net.

INVITED TALK

- ***Warehouse-scale Computers***

  Luiz André Barroso, Google Inc.

  Summarized by Minas Gjoka (mgjoka@uci.edu)

Luiz André Barroso said he intended to describe the characteristics of warehouse-scale computing infrastructure at Google from the hardware standpoint. Nowadays, warehouses are becoming more cost-effective for many companies, and in the near future new technology advancements may produce machines that will have properties that need to be tackled in today's warehouse-sized computers, such as thread concurrency, power saving, complexity management, and fault handling. Thousands of programs in different machines should work as a reliable platform running different services.

The first topic discussed was the programming efficiency for such systems. The need for parallelism to handle large amounts of data, heterogeneity, and failure-prone components complicates programming. A single programming system or language may not be enough. Instead Luiz advocated that the solution should be higher-level and use-specialized building blocks for large-scale distributed systems such as MapReduce and BigTable.

When building fault-tolerant software, it is important to guarantee that system interruption does not occur; otherwise some of the worst performance problems may appear. Service-level measurements that monitor performance provide only a partial view. Instead, Google has built a System Health infrastructure that collects health signals from all its servers and stores these signals perpetually in time series. Using this infrastructure, an analysis of hard disk failures was performed out of detailed signals collected during a period of nine months from a five-year inventory database. Understanding when such failures occur should, ideally, give a prediction model for failures that would allow preemptive action. (You can learn more about this project by reading the Pinheiro et al. paper that appeared at FAST '07 or the summary that appeared in the June 2007 issue of *;login:*.)

Grouping disks by age did not give conclusive results, because of the different hard drive model mixtures in the data. An interesting finding is that temperature has little impact on the average failure rate. In fact, higher failure rates are observed at lower temperatures.

Signals were collected from the standard SMART interface of hard disks, in an effort to build a predictive failure model. The results showed that only a subset of the SMART signals are strong indicators of future failures. For example, drives with scan errors are ten times more likely to fail. However, the predictive power of SMART signals seems limited, since almost half of the failures appear unpredictable when the set of strong indicators is used.

The cost of operation, in terms of energy and maximization of utilization, needs to be taken into account for warehouse building. The former becomes even more important since, unlike hardware costs, energy prices are increasing. Luiz mentioned that energy costs (excluding cooling) can account for up to 20% of the company's IT budget. Part of the solution lies in improving the efficiency of power supplies, which ranges from 55% to 70% nowadays, by reducing conversion losses. It is easy to see that with 55% efficiency the power supply becomes the largest power consumer inside a machine.

The goal of the maximization of utilization is to maximize the facility usage without exceeding contractual capacity limits. A six-month power monitoring study was conducted at Google to examine opportunities in power subscriptions. The study included three machine aggregation levels (rack, power distribution unit, cluster), with each almost an order of magnitude larger then the previous one. The analysis showed that at the cluster level the normalized power never exceeded 71%, which leaves room for more servers to be packed in the warehouse.

Given that current machines usually consume around 50% of their peak rate at idle mode, simulations for potential improvements in power consumption behavior were performed. The idea was to assume the availability of active low-power modes (at most, 5% of peak power). The simulation results showed remarkable improvements for both peak power and energy at the cluster level. It was suggested that power-saving features be implemented for other components in addition to the CPU and that a wide dynamic power range with low consumption at idle mode be included.

In conclusion, Luiz reiterated the benefits of understanding failures and emphasized the potential for power and energy efficiency. Most questions wandered around the issue of power savings.

Information for the climate savers computing initiative referenced by the speaker can be found at http://www.climatesaverscomputing.org/.

PLENARY CLOSING SESSION

- ***Crossing the Digital Divide: The Latest Efforts from One Laptop per Child***

Mary Lou Jepsen, CTO, One Laptop per Child

Summarized by Rik Farrow (rik@usenix.org)

Mary Lou Jepson said that because she has been traveling so much, promoting One Laptop per Child (OLPC), it is hard for her to keep track of what time zone she presently inhabits. Some of her jetlag was apparent in her somewhat rambling talk, but I still found what she had to say fascinating. You can find transcripts of recent talks by Jepson at http://www.olpctalks.com/mary_lou_jepsen/ and more about the hardware of the current version of the laptop, the XO rev C, at http://wiki.laptop.org/go/Hardware_specification.

Jepson had been an engineer at Intel specializing in display technologies. She went to work for OLPC as it was realized that the single most expensive part of a PC that is designed to be cheap would be the display. She explained two innovative features of her display design which reduce cost and power while increasing usability. The first innovation is the creation of dual-mode display cells. Most LCD displays rely on backlighting that shines through a color filter, then through the liquid crystal cell, which is more or less transparent. That means the display focuses on chrominance, whereas the human eye is actually more sensitive to luminance. She designed a screen that has greater resolution in reflective (black-and-white) mode, 200 dots per inch, which makes the screen very easy to read. A significant motivation for this design, besides the use of less power than with backlighting, is that it can store and display textbooks. Instead of buying textbooks for children, countries can buy XOs and upload textbooks, which will pay for the laptop over its expected lifetime. When used with backlight, the display goes from 1200x900 mono to 800x600 color, and it may use as much as ten times as much power because of the backlighting.

The other innovation has to do with refreshing the screen while the CPU is idle. Normally, the CPU must copy data to refresh the display 30 times per second, but by replacing the ordinary display controller chip with one that has memory, the chip takes over refreshes for the CPU, allowing the CPU to sleep until needed.

The XO has other energy-saving innovations, such as the use of a Marvel WiFi chip that includes part of an ARM CPU. That means that a laptop can function as part of a mesh network even when it is otherwise idle, as the Marvel chip handles the processing. The storage is a 1024-MB NAND flash and 256 MB of RAM. Jepson also showed examples of robust servers designed to act as both additional storage and connections to external networks.

The onetime symbol of the laptop, the handcrank, has been replaced with swivel-up ears that contain the WiFi antennas. When swiveled down, the ears cover external connectors, such as USB ports. Power can come from solar cells or generators, including a salad-spinner design for hand-charging.

The display itself is gorgeous, and the system is certainly sturdy. Jepson explained that laptops at a test site in Nigeria were breaking after only three months (but had been expected to last five years). It turns out that the desktops at the test site are slanted, and the laptops were falling off the desks onto the concrete floors at the rate of several times an hour (and continued to work for three months!). The XO is designed so that it can be repaired locally by swapping out parts, and extra case screws are included to replace lost screws.

The laptop runs a version of Linux, and it uses Sugar as the GUI framework. Software is designed so that the laptop can be used by children who cannot yet read. Jepson told us of a project in India where children taught themselves how to read by having some access to computer displays.

The simple, rugged, low-power, and low-cost design makes the XO not just an ideal textbook replacement but a desirable product in other worldwide markets. I commented to Jepson that licensing the display and some of the other technology might be one way of supporting further development of the OLPC vision.

Other questioners had darker views of the project. Tristan Lawrence said that he expected that the laptops will never reach their intended recipients, suggesting that the governments will distribute the laptops to better-off city dwellers, rather than the apparent targets of the project. Jepson answered that they have focused on teaching and designing a low-power laptop with mesh networking and a usable display, not on politics. She did mention Bitfrost, the security used to help prevent theft of laptops. Laptops must be updated with a signed key once every several weeks, or they will stop working. The laptops can also be remotely disabled if stolen. For more on Bitfrost and the security model of OLPC, see http://lwn.net/Articles/221052/.

Aarjav Trivedi of Secure Computing said that he is from India, a country that has so far decided not to buy the laptop. Trivedi declared that content is key and wondered if they had found local content in India. Jepson said that they have been working with local people to scan books. Even though India and China have been cool to the project so far, half the children in the world live in China and India.

Quando Lee asked about textbook cost comparisons. Jepson answered that, for example, in Brazil, textbooks cost $20/year, so over its expected five-year lifetime, the XO pays for itself. Experience in China showed that kids allowed to read anything they wanted learned five times as many Chinese characters as other children with less to read. The same person said that textbooks can last more than five years and that he had used his older brother's books. Jepson responded by saying that textbooks cost $643 per year in Massachusetts.

Marc Fuscinski said that he had visited some site in Sao Paolo, Brazil, but the kids can't take the laptops home. Jepson says that is certainly true. But in other places kids are starting a "right to laptop" movement, and in Cambodia, Thailand, and Nigeria they can take them home.

Someone wondered why the laptop couldn't last more than five years. Jepson replied that the LCD will last for half a million hours in sunlight, but the flash memory has a limited number of write cycles (because of wear leveling). The same person asked whether the laptop is recyclable, and Jepson answered that if a laptop stops working, it can be given to a post office, where it will be sent to a central depot for repair or recycling. The entire device is green, and it costs more to ship it than to recycle it.

George Herbert of Open Software Foundation asked whether the OLPC planned on frontloading open content. Jepson's understanding is that the countries will choose what content they want on the server. For example, they ask the participating country to pick the top 100 books appropriate for kids to read, while they provide a Mathematica lite version, along with reading and drawing tools. Kids can also program the computers themselves, using Python or Logo.

Warren Henson of Google asked about other plans for long-term storage and backup. Jepson said that that sounds like a great thing for Google to do. Right now they are stuck with the server (a low-power device, sealed with a hard disk) and aren't currently addressing that problem, although Google has provided gmail accounts. In response to Henson's mention of alternative projects from other organizations, Jepson said that they would like to work with these groups and try every week. When these efforts fail, the kids lose, in her opinion. Since Jepson spoke, Intel has announced that they plan to work with OLPC, and Intel is now listed as a supporter on the laptop.org site.

Linux Symposium 2007

Ottawa, Canada
June 27–30, 2007
Summarized by Rick Leir (rickleir@leirtech.com)

OLS is the conference for Linux kernel programmers, across the spectrum from embedded to large SMP systems. It also attracts application programmers and systems admins. Last year it was co-located with the Kernel Summit, but not this year. Attendees came from around the world. For me, travel arrangements were simple: The express city bus is convenient to me!

There were three tracks concurrent with tutorials, and several times I wanted to be in two places at a time. For a complete schedule see www.linuxsymposium.org/2007/. For a more detailed summary see www.linux.com/feature/115608. The attendees included a few hobbyists and academics, but most people were from companies including IBM, Intel, Sony, Red Hat, and AMD.

Jon Corbet gave his yearly Kernel Report. The trend is toward faster major releases. Where these used to be spaced by years, now they are spaced by months. The release cycle is more predictable than before, with a merge window of 2 weeks followed by 6 weeks for stabilization. This quickly moves changes out to users. Also, distributors (e.g., Red Hat, Ubuntu) are closer to the mainline. There is excellent tracking and merging of patches considering the volume (though some say quality was horrific for 2.6.21). There is ongoing work on automated testing.

For kernel 2.6.22, there will be:

- A new mac80211 wireless stack
- UBI flash-aware volume management
- An IVTV video tuner driver
- A new CFQ (Complete Fair Queueing) IO scheduler
- A firewire stack
- A SLUB memory management allocator (http://lwn.net/Articles/229984/)

In terms of scalability, SMP with 512 CPUs works well, and work on locks and page management processes for larger systems.

For filesystems, Jon observed that disks are getting larger but not faster, so fsck time can be a problem. New filesystems such as chunkfs and tilefs are more scalable and subdivide a disk so that only the active part needs to be fsck'd. The btrfs filesystem is extent-based, with subvolumes. It supports snapshot checksumming, online fsck, and faster fsck. Jon talked about ext4 with its 48-bit block numbers and use of extents. [Editor's note: Also see the article in the June 2007 issue of *;login:* about ext4.]

Jon finds reiser4 interesting, but the project is unfortunately stalled because Hans Reiser is no longer able to work on it. It needs a new champion.

Jon talked about virtualization. It is getting more attention, as shown by the many related presentations at this conference (e.g., KVM, lguest, Vserver).

The kernel is unlikely to go to GPL version 3 even if that was desired, because it is currently licensed with GPL version 2 and thousands of contributors would have to be contacted to make the change.

Jon's article has summaries of the Symposium and a summary of the work going into 2.6.22 (see http://lwn.net/Articles/240402/).

Greg Kroah-Hartman (www.kroah.com) taped to the back wall a 40-foot-long chart that linked together the people who have contributed patches. Developers were invited to sign the chart and about 100, of the 900 people who contributed to the 2.6.22 kernel, did so.

Mike Mason presented SystemTap, a dynamic tracing tool based on kprobes. Its simple scripting language provides a safe and flexible way to instrument a Linux system without modifying source code or rebooting.

There was considerable interest in embedded Linux. Robin Getz (blackfin.uclinux.org/) presented a tutorial on how to program for embedded systems with no MMU. I can't do uclinux justice here, but it seems to be the way to go.

Tim Chen talked about keeping kernel performance from regressions. He does weekly performance tests on the latest snapshot, and he occasionally sees large regressions. There are about 7000 patches per week, so it is not surprising that there would be problems. The 14 benchmarks include OLTP, an industry-standard Java business benchmark, cpu-int, cpu-fp, netperf, volanomark, lmbench, dbench, iozone, interbench, and httperf. The project is at kernel-perf.sourceforge.net/.

Arnaldo Carvalho de Melo talked about tools to help optimize kernel data structures. By rearranging the fields in structures you can avoid "holes" and thereby pack them into less memory. At times when related fields are close enough to be in the same cache line, performance improves. The pahole tool analyzes a struct and suggests field reordering.

Intel sponsored the reception Wednesday evening, and they demonstrated Ultra Wide Band (UWB) wireless networking (480 Mbps) between two laptops. Each was screening a video from the other's disk. This is a low-power technology to conserve battery power and avoid radio interference while being effective to 30 feet. UWB will be appearing in products soon.

Leonid Grossman talked about the challenges of 10Gb Ethernet. The transition to this is turning out to be more complex than earlier technology cycles. Part of the TCP stack is offloaded to the NIC TCP Offload Engine (TOE), so the kernel networking code has to change. I hear from osdl.org that there are significant problems with this.

Christopher James Lahey talked about Miro, which is a podcast client. He argues that "culture" is currently expressed via video, and we need a desktop app to search for video, display it, and organize channel folders or playlists. Miro uses Python, Pyrex, Javascript, CSS, and DOM. It uses some interesting database concepts. See more at getmiro.com.

Arnd Bergman of IBM talked about the Cell Broadband Engine, which is in Sony PS3 and IBM blades. This processor has the PowerPC Architecture with an L2 cache of 512 KB and 8 SPUs. The SPU is a co-processor that does fast floating-point math (though not so fast for double precision). There is a high-bandwidth bus (25 GB) connecting these processors, using explicit DMA. Each SPU has limited local memory (in effect, it executes out of its own cache), and overlay programming is used. Gcc emits DMA requests. Arnd evaluated the pros and cons of this package; on balance, it comes out very well.

From other sources I hear that the Sony PS3's Linux support involves a hypervisor that permits Linux to see only 6 of the SPUs. The NVIDIA video hardware is partly off limits to Linux programmers. Sony wants to interest the Open Source community in its products and very generously gave away several PS3s.

Andrew Cagney talked about Frysk, which is a user-level debugging tool for C and C++. It appears (my impression) to be as useful as Eclipse while being of considerably lighter weight. He talked about test-driven development and described a kernel regression test suite that has been discovering recent kernel bugs.

Jordan H. Crouse talked about using LinuxBIOS to speed up boot times and provide a more friendly boot environment. Be careful loading your motherboard flash memory.

Rusty Russell (ozlabs.org) entertained us with lguest, his simple virtualization project. He talked about how he went about coding lguest. He requires the guest OS to be the same version of Linux as the host, and his system does not support many of the features touted by the other virtual server systems, thereby saving much effort. He has my support!

Marcel Holtmann (bluez.org) talked about the latest Bluetooth tools and integration with Linux D-Bus. There was lots to talk about here, whether you are interested in the desktop or embedded devices.

Peter Zijlstra talked about the pagecache lock, which is not scalable. He has a way to avoid using a lock here. He alters the radix tree in order to support concurrent modifications of the data structure.

Jon "Maddog" Hall's ending keynote covered the Linux Terminal Server Project (LTSP.org) and how it could benefit poorer communities in the developing world. He made a convincing argument that this project was more practical than the One Laptop per Child (OLPC) project, and he showed a photo of himself in a school in Brazil. The terminals are not the VT200 of yore, but diskless Linux systems.

writing for ;login:

Writing is not easy for most of us. Having your writing rejected, for any reason, is no fun at all. The way to get your articles published in *;login:*, with the least effort on your part and on the part of the staff of ;login:, is to submit a proposal first.

PROPOSALS

In the world of publishing, writing a proposal is nothing new. If you plan on writing a book, you need to write one chapter, a proposed table of contents, and the proposal itself and send the package to a book publisher. Writing the entire book first is asking for rejection, unless you are a well-known, popular writer.

;login: proposals are not like paper submission abstracts. We are not asking you to write a draft of the article as the proposal, but instead to describe the article you wish to write. There are some elements that you will want to include in any proposal:

- What's the topic of the article?
- What type of article is it (case study, tutorial, editorial, mini-paper, etc.)?
- Who is the intended audience (syadmins, programmers, security wonks, network admins, etc.)?
- Why does this article need to be read?
- What, if any, non-text elements (illustrations, code, diagrams, etc.) will be included?
- What is the approximate length of the article?

Start out by answering each of those six questions. In answering the question about length, bear in mind that a page in *;login:* is about 600 words. It is unusual for us to publish a one-page article or one over eight pages in length, but it can happen, and it will, if your article deserves it. We suggest, however, that you try to keep your article between two and five pages, as this matches the attention span of many people.

The answer to the question about why the article needs to be read is the place to wax enthusiastic. We do not want marketing, but your most eloquent explanation of why this article is important to the readership of *;login:*, which is also the membership of USENIX.

UNACCEPTABLE ARTICLES

;login: will not publish certain articles. These include but are not limited to:

- Previously published articles. A piece that has appeared on your own Web server but not been posted to USENET or slashdot is not considered to have been published.
- Marketing pieces of any type. We don't accept articles about products. "Marketing" does not include being enthusiastic about a new tool or software that you can download for free, and you are encouraged to write case studies of hardware or software that you helped install and configure, as long as you are not affiliated with or paid by the company you are writing about.
- Personal attacks

FORMAT

The initial reading of your article will be done by people using UNIX systems. Later phases involve Macs, but please send us text/plain formatted documents for the proposal. Send proposals to login@usenix.org.

DEADLINES

For our publishing deadlines, including the time you can expect to be asked to read proofs of your article, see the online schedule at http://www.usenix .org/publications/login/sched .html.

COPYRIGHT

You own the copyright to your work and grant USENIX permission to publish it in ;login: and on the Web. USENIX owns the copyright on the collection that is each issue of *;login:*. You have control over who may reprint your text; financial negotiations are a private matter between you and any reprinter.

FOCUS ISSUES

In the past, there has been only one focus issue per year, the December Security edition. In the future, each issue may have one or more suggested focuses, tied either to events that will happen soon after *;login:* has been delivered or events that are summarized in that edition.

Announcement and Call for Papers **USENIX**

2008 USENIX Annual Technical Conference

Sponsored by USENIX, The Advanced Computing Systems Association

http://www.usenix.org/usenix08

June 22–27, 2008 **Boston, MA, USA**

Important Dates

Paper submissions due: *Monday, January 7, 2008, 11:59 p.m PST (hard deadline)*
Invited talk proposals due: *Friday, February 1, 2008*
Notification to authors: *Wednesday, March 12, 2008*
Final papers due: *Tuesday, April 29, 2008*
Poster submissions due: *Tuesday, May 6, 2008*

Program Committee

Program Co-Chairs
Rebecca Isaacs, *Microsoft Research*
Yuanyuan Zhou, *University of Illinois at Urbana-Champaign*

Program Committee
Frank Bellosa, *University of Karlsruhe, Germany*
Jeff Chase, *Duke University*
Dawson Engler, *Stanford University*
Jason Flinn, *University of Michigan*
Keir Fraser, *University of Cambridge*
Steve Gribble, *University of Washington*
Liviu Iftode, *Rutgers University*
Arkady Kanevsky, *Network Appliance*
Angelos Keromytis, *Columbia University*
Emre Kıcıman, *Microsoft Research*
Sam King, *University of Illinois at Urbana-Champaign*
Jeff Mogul, *HP Labs*
Erich Nahum, *IBM T.J. Watson Research Center*
David Presotto, *Google*
Sean Rhea, *Intel Research Berkeley*
Erik Riedel, *Seagate Research*
Timothy Roscoe, *ETH Zürich*
Mike Swift, *University of Wisconsin*
John Wilkes, *HP Labs*
Emmett Witchel, *University of Texas*
Xiaolan (Catherine) Zhang, *IBM T.J. Watson Research Center*
Zheng Zhang, *Microsoft Research*

Poster Session Chairs
Emre Kıcıman, *Microsoft Research*
Sam King, *University of Illinois at Urbana-Champaign*

Overview

Authors are invited to submit original and innovative papers to the Refereed Papers Track of the 2008 USENIX Annual Technical Conference. We seek high-quality submissions that further the knowledge and understanding of modern computing systems, with an emphasis on implementations and experimental results. We encourage papers that break new ground or present insightful results based on practical experience. The USENIX conference has a broad scope; specific topics of interest include but are not limited to:

- Architectural interaction
- Deployment experience
- Distributed and parallel systems
- Embedded systems
- Energy/power management
- File and storage systems
- Networking and network services
- Operating systems
- Reliability, availability, and scalability
- Security, privacy, and trust
- System and network management and troubleshooting
- Usage studies and workload characterization
- Virtualization
- Web technology
- Wireless, sensor, and mobile systems

Best Paper Awards

Cash prizes will be awarded to the best papers at the conference. Please see the USENIX Compendium of Best Papers for examples of Best Papers from previous years: http://www.usenix.org/publications/library/proceedings/best_papers.html.

How to Submit

Authors are required to submit full papers by 11:59 p.m. PST, Monday, January 7, 2008. *This is a hard deadline; no extensions will be given.*

All submissions for USENIX '08 will be electronic, in PDF format, through the conference Web site. USENIX '08 will accept two types of papers:

- **Regular Full Papers:** Submitted papers must be no longer than 14 single-spaced pages, including figures, tables, and references, using 10 point font. The first page of the paper should include the paper title and author name(s); reviewing is single-blind. Papers longer than 14 pages will not be reviewed. In a good paper, the authors will have:

- Attacked a significant problem
- Devised an interesting and practical solution
- Clearly described what they have and have not implemented
- Demonstrated the benefits of their solution
- Articulated the advances beyond previous work
- Drawn appropriate conclusions

◆ **Short Papers:** Authors with a contribution for which a full paper is not appropriate may submit short papers of at most 6 pages, applying the same formatting guidelines. Examples of short paper contributions include:

- Original or unconventional ideas at a preliminary stage of development
- The presentation of interesting results that do not require a full-length paper, such as negative results or experimental validation
- Advocacy of a controversial position or fresh approach

Accepted short papers will be published in the Proceedings and included in the Poster Session, and time will be provided for brief presentations of these papers.

Specific questions about submissions may be sent to usenix08chairs@usenix.org.

Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, program committees may share information about submitted papers with other conference chairs and journal editors to ensure the integrity of papers under consideration. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case.

Note that the above does not preclude the submission of a regular full paper that overlaps with a previous short paper or workshop paper. However, any submission that derives from an earlier workshop paper must provide a significant new contribution, for example, by providing a more complete evaluation.

Authors uncertain whether their submission meets USENIX's guidelines should contact the program chairs, usenix08chairs@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

Papers accompanied by nondisclosure agreements cannot be accepted. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors will be notified of paper acceptance or rejection by Wednesday, March 12, 2008. Accepted papers may be shepherded by a program committee member. Final papers must be no longer than 14 pages, formatted in 2 columns, using 10 point Times Roman type on 12 point leading, in a text block of 6.5" by 9".

Note regarding registration: One author per accepted paper will receive a registration discount of $200. USENIX will offer a complimentary registration upon request.

Poster Session

The poster session, held in conjunction with a reception, will allow researchers to present recent and ongoing projects. The poster session is an excellent forum to discuss new ideas and get useful feedback from the community. The poster submissions should include a brief description of the research idea(s); the submission must not exceed 2 pages. Accepted posters will be put on the conference Web site; however, they will not be printed in the conference Proceedings. Send poster submissions to session chairs Emre Kıcıman and Sam King at usenix08posters@usenix.org by Tuesday, May 6, 2008.

Birds-of-a-Feather Sessions (BoFs)

Birds-of-a-Feather sessions (BoFs) are informal gatherings organized by attendees interested in a particular topic. BoFs will be held in the evening. BoFs may be scheduled in advance by emailing bofs@usenix.org. BoFs may also be scheduled at the conference.

Invited Talks

These survey-style talks given by experts range over many interesting and timely topics. The Invited Talks track also may include panel presentations and selections from the best presentations at recent USENIX conferences.

The Invited Talks Committee welcomes suggestions for topics and requests proposals for particular talks. In your proposal, state the main focus, including a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit proposals via email to usenix08it@usenix.org by Friday, February 1, 2008.

Training Program

USENIX's highly respected training program offers intensive, immediately applicable tutorials on topics essential to the use, development, and administration of advanced computing systems. Skilled instructors, hands-on experts in their topic areas, present both introductory and advanced tutorials.

To provide the best possible tutorial slate, USENIX continually solicits proposals for new tutorials. If you are interested in presenting a tutorial, contact Dan Klein, Training Program Coordinator, tutorials@usenix.org.

Program and Registration Information

Complete program and registration information will be available in March 2008 on the USENIX '08 Web site. If you would like to receive the latest USENIX conference information, please join our mailing list at http://www.usenix.org/about/mailing.html.

Last Updated: 8/3/07

PROFESSORS, CAMPUS STAFF, AND STUDENTS— DO YOU HAVE A USENIX REPRESENTATIVE ON YOUR CAMPUS? IF NOT, USENIX IS INTERESTED IN HAVING ONE!

The USENIX Campus Rep Program is a network of representatives at campuses around the world who provide Association information to students, and encourage student involvement in USENIX. This is a volunteer program, for which USENIX is always looking for academics to participate. The program is designed for faculty who directly interact with students. We fund one representative from a campus at a time. In return for service as a campus representative, we offer a complimentary membership and other benefits.

A campus rep's responsibilities include:

- Maintaining a library (online and in print) of USENIX publications at your university for student use
- Distributing calls for papers and upcoming event brochures, and re-distributing informational emails from USENIX
- Encouraging students to apply for travel grants to conferences
- Providing students who wish to join USENIX with information and applications
- Helping students to submit research papers to relevant USENIX conferences
- Providing USENIX with feedback and suggestions on how the organization can better serve students

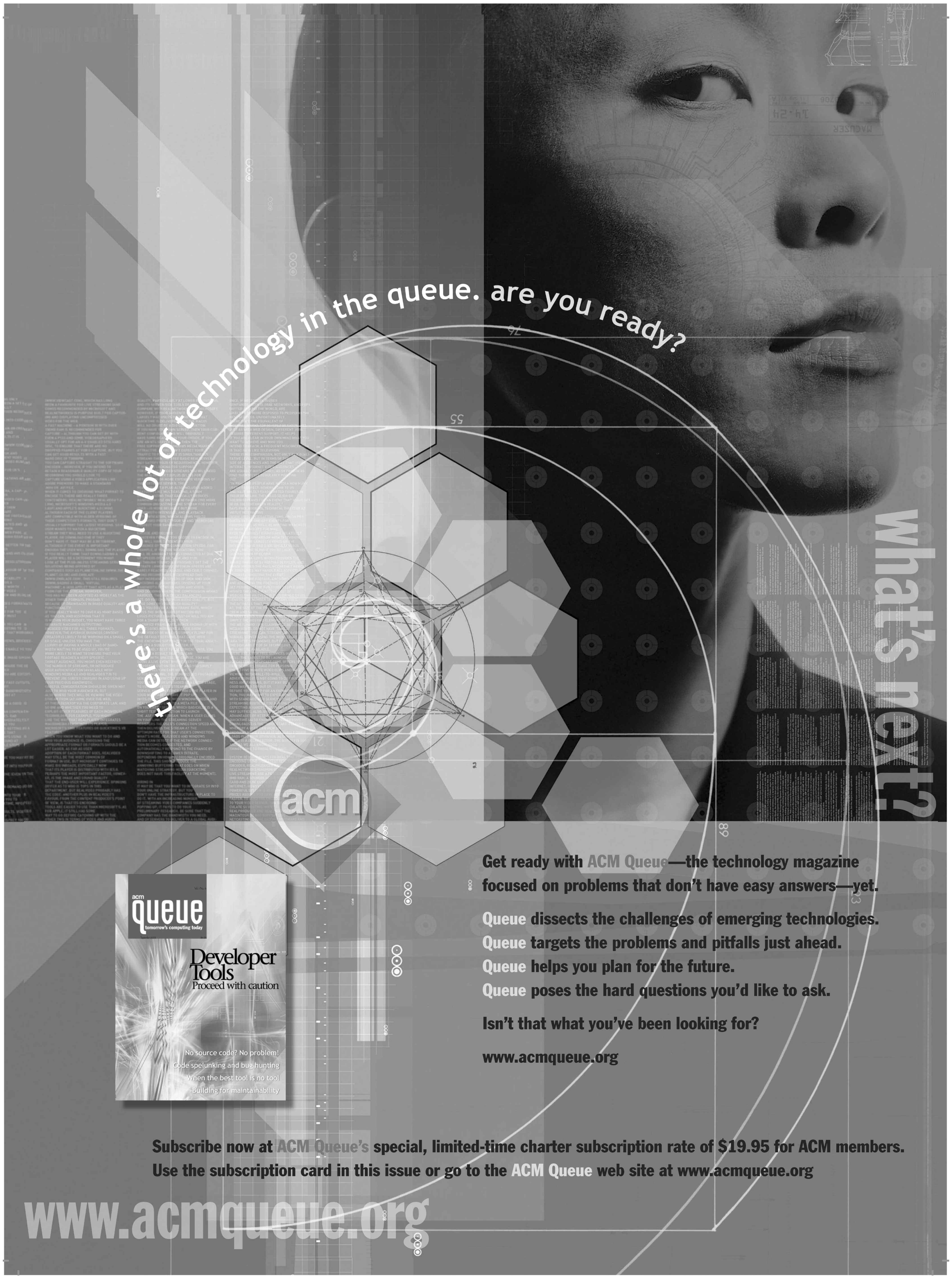
In return for being our "eyes and ears" on campus, representatives receive a complimentary membership in USENIX with all membership benefits (except voting rights), and a free conference registration once a year (after one full year of service as a campus rep).

To qualify as a campus representative, you must:

- Be full-time faculty or staff at a four year accredited university
- Have been a dues-paying member of USENIX for at least one full year in the past

For more information about our Student Programs, see http://www.usenix.org/students

USENIX contact: Anne Dickison, Director of Marketing, anne@usenix.org



there's a whole lot of technology in the queue. are you ready?
what's next?
acm
acm queue tomorrow's computing today
Developer Tools
Proceed with caution
No source code? No problem!
Code spelunking and bug hunting
When the best tool is no tool
Building for maintainability

Get ready with ACM Queue—the technology magazine focused on problems that don't have easy answers—yet.

Queue dissects the challenges of emerging technologies.
Queue targets the problems and pitfalls just ahead.
Queue helps you plan for the future.
Queue poses the hard questions you'd like to ask.

Isn't that what you've been looking for?

www.acmqueue.org

Subscribe now at ACM Queue's special, limited-time charter subscription rate of $19.95 for ACM members. Use the subscription card in this issue or go to the ACM Queue web site at www.acmqueue.org

www.acmqueue.org

LISA '07

Sponsored by USENIX and SAGE

21ST LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE

November 11–16, 2007, Dallas, TX

- 6 days of training by experts in their fields
- 3-day technical program
  - Keynote Address by John Strassner, *Vice President, Autonomic Networking and Communications, Motorola Research Labs*
  - Invited talks by industry leaders
  - Refereed Papers, Hit the Ground Running Track, Guru Is In Sessions, Workshops, BoFs, WiPs, Posters, and more!
- Vendor Exhibition
- And more!

Online registration
is now open at
www.usenix.org/lisa07

Register by October 19, 2007, and save!

http://www.usenix.org/lisa07

;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