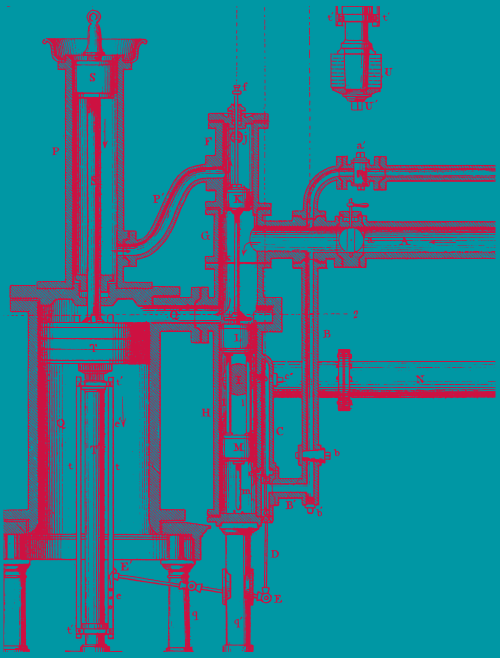
FEBRUARY 2008 VOLUME 33 NUMBER 1

;login:

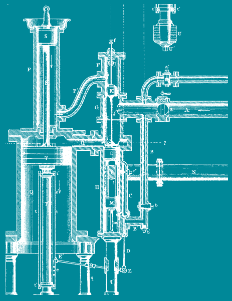
THE USENIX MAGAZINE



USENIX

The Advanced Computing Sytems Association

USENIX Upcoming Events



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

Sponsored by ACM SIGPLAN in cooperation with USENIX

MARCH 5–7, 2008, SEATTLE, WA, USA
http://vee08.cs.tcd.ie

Usability, Psychology, and Security 2008

Co-located with NSDI '08

APRIL 14, 2008, SAN FRANCISCO, CA, USA
http://www.usenix.org/upsec08

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

Botnets, Spyware, Worms, and More

Co-located with NSDI '08

APRIL 15, 2008, SAN FRANCISCO, CA, USA
http://www.usenix.org/leet08

Workshop on Organizing Workshops, Conferences, and Symposia for Computer Systems (WOWCS '08)

Co-located with NSDI '08

APRIL 15, 2008, SAN FRANCISCO, CA, USA
http://www.usenix.org/wowcs08

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

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 16–18, 2008, SAN FRANCISCO, CA, USA
http://www.usenix.org/nsdi08

The Sixth International Conference on Mobile Systems, Applications, and Services (MobiSys 2008)

Jointly sponsored by ACM SIGMOBILE and USENIX

JUNE 10–13, 2008, BRECKENRIDGE, CO, USA
http://www.sigmobile.org/mobisys/2008/

2008 USENIX Annual Technical Conference

JUNE 22–27, 2008, BOSTON, MA, USA
http://www.usenix.org/usenix08

2nd International Conference on Distributed Event-Based Systems (DEBS 2008)

Organized in cooperation with USENIX, the IEEE and IEEE Computer Society, and ACM (SIGSOFT)

JULY 2–4, 2008, ROME, ITALY
http://debs08.dis.uniroma1.it/
Abstract submissions due: March 9, 2008

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

Co-located with USENIX Security '08

JULY 28–29, 2008, SAN JOSE, CA, USA
http://www.usenix.org/evt08
Paper submissions due: March 28, 2008

2nd USENIX Workshop on Offensive Technologies (WOOT '08)

Co-located with USENIX Security '08

JULY 28, 2008, SAN JOSE, CA, USA

17th USENIX Security Symposium

JULY 28–AUGUST 1, 2008, SAN JOSE, CA, USA
http://www.usenix.org/sec08

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

Co-located with USENIX Security '08

JULY 29, 2008, SAN JOSE, CA, USA
http://www.usenix.org/hotsec08
Position paper submissions due: May 28, 2008

22nd Large Installation System Administration Conference (LISA '08)

Sponsored by USENIX and SAGE

NOVEMBER 9–14, 2008, SAN DIEGO, CA, USA
http://www.usenix.org/lisa08
Extended abstract and paper submissions due: May 8, 2008

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

Sponsored by USENIX in cooperation with ACM SIGOPS

DECEMBER 8–10, 2008, SAN DIEGO, CA, USA
http://www.usenix.org/osdi08
Paper submissions due: May 8, 2008

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

contents

;login:

VOL. 33, #1, FEBRUARY 2008

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
David Couzens
proofshop@usenix.org

PRODUCTION
Casey Henderson
Michele Nelson

TYPESETTER
Star Type
startype@comcast.net

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

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

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

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

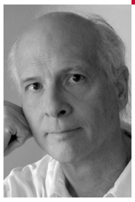
$85 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $120 per year.

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

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

RIK FARROW

musings



rik@usenix.org

THERE ARE TIMES WHEN WE JUST can't wait for the future to arrive, such as the coming of warmer weather. And sometimes it seems that people pine for the poorly remembered past, as if it were somehow better than what we face today. Right now, I want to talk about sysadmins and ponder whether they are looking ahead while wishing for an imagined past.

In this issue you will find the summaries for LISA '07, including the summary I wrote about John Strassner's keynote. John spoke about experiences with a project at Motorola where researchers had created a functioning example of network autonomics. This is a complex system, with many different active components all contributing to decisions that result in changes in configuration. The FOCALE architecture (see slide 23 of his presentation on the LISA '07 page [1]) has a Context Manager, a Policy Manager, and an Autonomic Manager, as well as a machine learning component, all of which are involved in controlling the creation and modifications of device configurations.

FOCALE is a working system. It actually helps to simplify a terribly complex control setup that includes seven different groups of administrators (see slide 4). John carefully began his talk by explaining the existing situation found in many telecommunications companies (think cell phone operators). He explained the limitations of the current network management, including the need for human involvement in analysis before anything can be done. And he described what he means by autonomics, going way behind the infamous four self-functions of self-configuration, self-protection, self-healing, and self-optimization made famous by IBM [2, 3]. John considers these benefits, seeing the way forward via knowledge about component systems, the context in which they operate, and an ability to learn and reason, to follow policy determined from business rules, and to adapt offered services and resources as necessary.

I thought John's talk described groundbreaking research, where a real autonomic system was working to make a network function more smoothly. But others at the conference weren't nearly as happy. The most common complaint, one that really stuck with me, was that there was "too much math" in his solution. I wondered whether the *two* equations found on slide 47 (shades of calculus!) were to blame. But then I read Alva Couch's article (page 12) and realized that perhaps the real problem was something completely different. The real problem has to do with two things: a mindset, and being stuck in the past.

The Mindset

Alva Couch explains something I have had difficulty understanding since I first encountered the concept, way back when I was a college student. I found languages such as FORTRAN and ALGOL easy to comprehend, but LISP and APL unpleasant to use. I've recently learned that there is a "semantic wall" between these two languages, to borrow from Alva. Using a more modern example, the C language is bottom-up, or imperative. You write a sequence of commands, and they are executed in order. Functional programming languages, such as LISP or Haskell, work top-down, where the entire program is a single expression. In a functional language, the expression describes the desired result without specifying how the result is derived.

Now consider how system administration gets accomplished today. Someone requests a change to a service, and sysadmins go about changing the configuration, an imperative operation. If something breaks, the sysadmins set about uncovering the cause of the problem and adjusting the configuration to solve the problem—again, a bottom-up approach.

What John Strassner, and Alva Couch, suggest requires a mindset that is very different. Instead of acting imperatively, getting right into the nitty-gritty of configuration editing, autonomics requires a more functional, top-down approach. I believe that a lot of sysadmins will find this approach inimical to the way they have carried out their duties for their entire working careers.

And thus the past, in which we do things the way they have always been done, becomes an obstacle to a future where some things will need to be done differently. We are really not that different from people riding horse-drawn carriages in 1907 complaining about the noisy and dangerous horseless carriages.

Autonomic computing does not mean the end of understanding and editing configuration files. It will mean that this task will consume less of the sysadmin's working day. I expect autonomics, in some form, will evolve, regardless of kicking, screaming, and temper tantrums or editorializing against its adoption. And the people who develop autonomics may not be sysadmins but researchers willing to take a top-down, instead of a bottom-up, approach.

So times change: either the world becomes more complicated, or it appears more complicated because it now works differently. Remember, there are still many people living in developed countries who do not use electronic communication such as email, IM, and text messaging. Don't get left behind.

The Lineup

I had often wondered about anycasting, so I contacted ISC and found Joe Abley willing and able to describe the pros, cons, and sheer aggravation surrounding the use of anycasting in IPv4. Anycasting is not a solution that many can use, but I believe you should be aware of it.

Alva Couch follows with his article that examines mindsets, or the semantic wall I also attempted to describe in this editorial. Learning more about the differences between imperative and functional programming languages is not the point of Alva's article; the example is just used to demonstrate what he considers the crux of building real autonomic computing systems.

Eric Langheinrich next tells us about a method for controlling access to Web content that relates to anti-spam techniques. Through the use of scripts and a scoring service, you can configure your Web server to deny content to crawlers looking for certain content, such as email addresses, to be used in later UCE.

We next have two articles related to papers presented at LISA '07. A group from the University of Arizona describes Stork, a package management system designed for use in clusters and PlanetLab. Once you have installed your distributed applications, you can consider managing those applications using Plush, the second in this set of articles.

Octave Orgeron then continues his tutorial on Solaris LDoms, with a focus on advanced topics in network and disk configuration. He is followed by Aditya Sood, who explains problems with XML signing.

We have a new columnist starting with this issue of *;login:*. Peter Galvin, longtime tutorial instructor at USENIX conferences as well as the Solaris columnist for the now-defunct *Sys Admin*, has agreed to write about Solaris for *;login:*. I am happy to help provide a new home for Pete's column and hope that many of you will continue to enjoy reading it.

And, as mentioned, we have summaries of LISA '07, as well as of four of the workshops that occurred before the main conference began.

Starting with this issue, *;login:* will include a cartoon courtesy of User-Friendly. We are thankful to David Barton for allowing us to lighten up our pages with some relevant humor.

Times are changing. But then times always change, and those changes often prove upsetting and difficult even to consider, much less accommodate. What sysadmins face today is an exploding number of computers, and computer-enabled devices, that must be managed. We need to look toward new technologies that will make managing these devices easier, even if the transition will be difficult. And I can't imagine it will be easy.

REFERENCES

[1] LISA '07 Technical Sessions: http://www.usenix.org/events/lisa07/tech/.

[2] Home page for IBM's Autonomics project: http://www.research.ibm.com/autonomic/index.html.

[3] Wikipedia page, with more links about autonomics at the bottom: http://en.wikipedia.org/wiki/Autonomic_Computing.

JOE ABLEY

fear and loathing in the routing system

Joe Abley is the Director of Operations at Afilias Canada, a DNS registry company, and a technical volunteer at Internet Systems Consortium. He likes his coffee short, strong, and black and is profoundly wary of outdoor temperatures that exceed 20°C.

jabley@ca.afilias.info

ANYCAST IS A STRANGE ANIMAL. IN some circles the merest mention of the word can leave you drenched in bile; in others it's an overused buzzword which triggers involuntary rolling of the eyes. It's a technique, or perhaps a tool, or maybe a revolting subversion of all that is good in the world. It is "here to stay." It is by turns "useful" and "harmful"; it "improves service stability," "protects against denial-of-service attacks," and "is fundamentally incompatible with any service that uses TCP."

That a dry and, frankly, relatively trivial routing trick could engender this degree of emotional outpouring will be unsurprising to those who have worked in systems or network engineering roles for longer than about six minutes. The violently divergent opinions are an indication that context matters with anycast more than might be immediately apparent, and since anycast presents a very general solution to a large and varied set of potential problems, this is perhaps to be expected.

The trick to understanding anycast is to concentrate less on the "how" and far more on the "why" and "when." But before we get to that, let's start with a brief primer. Those who are feeling a need to roll their eyes already can go and wait outside. I'll call you when this bit is done.

Nuts and Bolts

Think of a network service which is bound to a greater extent than you'd quite like to an IP address rather than a name. DNS and NTP servers are good examples, if you're struggling to paint the mental image. Renumbering servers is an irritating process at the best of times, but if your clients almost always make reference to those servers using hard-coded IP addresses instead of names, the pain is far greater.

Before the average administrator has acquired even a small handful of battle scars from dealing with such services, it's fairly common for the services to be detached from the physical servers that house them. If you can point NTP traffic for 204.152.184.72 at any server you feel like, moving the corresponding service around as individual servers come and go becomes trivially easy. The IP address in this case becomes an identifier, like a DNS name, detached from the address of the server that happens to be running the server processes on this particular afternoon.

With this separation between service address and server address, a smooth transition of this NTP service from server A to server B within the same network is possible with minimal downtime to clients. The steps are:

1. Make sure the service running on both servers is identical. In the case of an NTP service, that means that both machines are running appropriate NTP software and that their clocks are properly synchronized.
2. Add a route to send traffic with destination address 204.152.184.72 toward server B.
3. Remove the route that is sending traffic toward server A.

Ta-da! Transition complete. Clients didn't notice. No need for a maintenance window. Knowing smiles and thoughtful nodding all round.

To understand how this has any relevance to the subject at hand, let's insert another step into this process:

2.5. Become distracted by a particularly inflammatory slashdot comment, spend the rest of the day grumbling about the lamentable state of the server budget for Q4, and leave the office at 11 p.m. as usual, forgetting all about step 3.

The curious result here is that the end result might very well be the same: Clients didn't notice. There is no real need for a maintenance window. What's more, we can now remove either one of those static routes and turn off the corresponding server, and clients *still* won't notice. We have distributed the NTP service across two origin servers using anycast. And we didn't even break a sweat!

Why does this work? Well, a query packet sent to a destination address arrives at a server which is configured to accept and process that query, and the server answers. Each server is configured to reply, and the source address used each time is the service address. The fact that there is more than one server available doesn't actually matter. To the client (and, in fact, to each server), it looks like there *is* only one server. The query-response behavior is exactly as it was without anycast on the client and on the server. The only difference is that the routing system has more than one choice about toward which server to send the request packet.

(To those in the audience who are getting a little agitated about my use of a stateless, single-packet exchange as an example here, there is no need to fret. I'll be pointing out the flies in the ointment very soon.)

The ability to remove a dependency on a single server for a service is very attractive to most system administrators, since once the coupling between service and server has been loosened, intrusive server maintenance without notice (and within normal working hours) suddenly becomes a distinct possibility. Adding extra server capacity during times of high service traffic without downtime is a useful capability, as is the ability to add additional servers.

For these kinds of transitions to be automatic, the interaction between the routing system and the servers needs to be dynamic: that is, a server needs to be able to tell the routing system when it is ready to receive traffic destined for a particular service, and correspondingly it also needs to be able to tell the routing system when that traffic should stop. This signaling can be made to work directly between a server and a router using standard routing protocols, as described in ISC-TN-2004-1 [1] (also presented at USENIX '04 [2]). This approach can also be combined with load balancers (sometimes called “layer-4 switches”) if the idea of servers participating in routing protocols directly is distasteful for local policy reasons.

This technique can be used to build a cluster of servers in a single location to provide a particular service, or to distribute a service across servers that are widely distributed throughout your network, or both. With a little extra attention paid to addressing, it can also be used to distribute a single service around the Internet, as described in ISC-TN-2003-1 [3].

Anycast Marketing

Some of the benefits to the system administrator of distributing a service using anycast have already been mentioned. However, making the lives of system administrators easier rarely tops anybody’s quarterly objectives, much as you might wish otherwise. If anycast doesn’t make the service better in some way, there’s little opportunity to balance the cost of doing it.

So what are the tangible synergies? What benefits can we whiteboard proactively, moving forward? Where are the bullet points? Do you like my tie? It’s new!

Distributing a service around a network has the potential to improve service availability, since the redundancy inherent in using multiple origin servers affords some protection from server failure. For a service that has bad failure characteristics (e.g., a service that many other systems depend on) this might be justification enough to get things moving.

Moving the origin server closer to the community of clients that use it has the potential to improve response times and to keep traffic off expensive wide-area links. There might also be opportunities to keep a service running in a part of your network that is afflicted by failures in wide-area links in a way that wouldn’t otherwise be possible.

For services deployed over the Internet, as well as nobody knowing whether you’re a dog, there’s the additional annoyance and cost of receiving all kinds of junk traffic that you didn’t ask for. Depending on how big a target you have painted on your forehead, the unwanted packets might be a constant drone of backscatter, or they might be a searing beam of laser-like pain that makes you cry like a baby. Either way, it’s traffic that you’d ideally like to sink as close to the source as possible, ideally over paths that are as cheap as possible. Anycast might well be your friend.

Flies in the Ointment

The architectural problem with anycast for use as a general-purpose service distribution mechanism results from the flagrant abuse of packet delivery semantics and addressing that the technique involves. It’s a hack, and as with any hack, it’s important to understand where the boundaries of normal operation are being stretched.

Most protocol exchanges between clients and servers on the Internet involve more than one packet being sent in each direction, and most also involve state being retained between subsequent packets on the server side. Take a normal TCP session establishment handshake, for example:

1. Client sends a SYN to a server.
2. Server receives the SYN and replies with a SYN-ACK.
3. Client receives the SYN-ACK and replies with an ACK.
4. Server receives the ACK, and the TCP session state on both client and server is “ESTABLISHED.”

This exchange relies on the fact that "server" is the same host throughout the exchange. If that assumption turns out to be wrong, then this happens:

1. Client sends a SYN to server A.
2. Server A receives the SYN and replies with a SYN-ACK.
3. Client receives the SYN-ACK and replies to the service address with an ACK.
4. Server B receives the ACK and discards it, because it has no corresponding session in "SYN-RECEIVED."

At the end of this exchange, the client is stuck in "SYN-SENT," server A is stuck in "SYN-RECEIVED," and server B has no session state at all. Clearly this does not satisfy the original goal of making things more robust; in fact, under even modest query load from perfectly legitimate clients, the view from the servers is remarkably similar to that of an incoming SYN flood.

It's reasonable to wonder what would cause packets to be split between servers in this way, because if that behavior can be prevented perhaps the original benefits of distributed services that gave us all those warm fuzzies can be realized without inadvertently causing our own clients to attack us. The answer lies in the slightly mysterious realm of routing.

The IP routing tables most familiar to system administrators are likely to be relatively brief and happily uncontaminated with complication. A single default route might well suffice for many hosts, for example; the minimal size of that routing table is a reflection of the trivial network topology in which the server is directly involved. If there's only one option for where to send a packet, that's the option you take. Easy.

Routers, however, are frequently deployed in much more complicated networks, and the decision about where to send any particular packet is correspondingly more involved. In particular, a router might find itself in a part of the network where there is more than one viable next hop toward which to send a packet; even with additional attributes attached to individual routes, allowing routers to prioritize one routing table entry over another, there remains the distinct possibility that a destination address might be reached equally well by following any one of several candidate routes. This situation calls for Equal-Cost Multi-Path (ECMP) routing.

Without anycast in the picture, so long as the packets ultimately arrive at the same destination, ECMP is probably no cause for lost sleep. If the destination address is anycast, however, there's the possibility that different candidate routes will lead to different servers, and therein lies the rub.

Horses for Courses

So, is anycast a suitable approach to making services more reliable? Well, yes and no. Maybe. Maybe not, too. Oh, it's all so vague! I crave certainty! And caffeine-rich beverages!

The core difficulty that leads to all this weak hand-waving is that it's very difficult to offer a general answer when the topology of even your own network depends on the perspective from which it is viewed. When you start considering internetworks such as, well, the Internet, the problem of formulating a useful general answer stops being simply hard and instead becomes intractable.

From an architectural perspective, the general answer is that for general-purpose services and protocols, anycast doesn't work. Although this is mathematically correct (in the sense that the general case must apply to all possible scenarios), it flies in the face of practical observations and hence doesn't really get us anywhere. Anycast is used today in applications ranging from the single-packet exchanges of the DNS protocol to multi-hour, streaming audio and video. So it does work, even though in the general case it can't possibly.

The fast path to sanity is to forget about neat, simple answers to general questions and concentrate instead on specifics. Just because anycast cannot claim to be generally applicable doesn't mean it doesn't have valid applications.

First, consider the low-hanging fruit. A service that involves a single-packet, stateless transaction is most likely ideally suited to distribution using anycast. Any amount of oscillation in the routing system between origin servers is irrelevant, because the protocol simply doesn't care which server processes each request, so long as it can get an answer.

The most straightforward example of a service that fits these criteria is DNS service using UDP transport. Since the overwhelming majority of DNS traffic on the Internet is carried over UDP, it's perhaps unsurprising to see anycast widely used by so many DNS server administrators.

As we move on to consider more complicated protocols—in particular, protocols that require state to be kept between successive packets—let's make our lives easy and restrict our imaginings to very simple networks whose behavior is well understood. If our goal is to ensure that successive packets within the same client-server exchange are carried between the same client and the same origin server for the duration of the transaction, there are some tools we can employ.

We can arrange for our network topology to be simple, such that multiple candidate paths to the same destination don't exist. The extent to which this is possible might well depend on more services than just yours, but then the topology also depends to a large extent on the angle you view it from. It's time to spend some time under the table, squinting at the wiring closet. (But perhaps wait until everybody else has gone home, first.)

We can choose ECMP algorithms on routers that have behavior consistent with what we're looking for. Cisco routers, for example, with CEF (Cisco Express Forwarding) turned on, will hash pertinent details of a packet's header and divide the answer space by the number of candidate routes available. Other vendors' routers have similar capabilities. If the computed hash is in the first half of the space, you choose the left-hand route; if the answer is in the other half, you choose the right-hand route. So long as the hash is computed over enough header variables (e.g., source address and port, destination address and port) the route chosen ought to be consistent for any particular conversation ("flow," in router-ese).

When it comes to deploying services using anycast across other people's networks (e.g., between far-flung corners of the Internet), there is little certainty in architecture, topology, or network design and we need instead to concentrate our thinking in terms of probability: We need to assess benefit in the context of risk.

> Internet, n: "the largest equivalence class in the reflexive transitive symmetric closure of the relationship 'can be reached by an IP packet from" (Seth Breidbart).

The world contains many hosts that consider themselves connected to the Internet. However, that "Internet" is different, in general, for every host—it's a simple truism that not all the nodes in the world that believe themselves to be part of "the" Internet can exchange packets with each other, and that's even without our considering the impact of packet filters and network address translation. The Internet is a giant, seething ball of misconfigured packet filters, routing loops, and black holes, and it's important to acknowledge this so that the risks of service deployment using anycast can be put into appropriate context.

A service that involves stateful, multi-packet exchanges between clients and servers on the Internet, deployed in a single location without anycast, will be unavailable for a certain proportion of hosts at any time. You can sometimes see signs of this in Web server and mail logs in the case of asymmetric failures (e.g., sessions that are initiated but never get established); other failure modes might relate to control failures (e.g., the unwise blanket denial of ICMP packets in firewalls which so often breaks Path MTU Discovery). In other cases the unavailability might have less mysterious origins, such as a failed circuit to a transit provider which leaves an ISP's clients only able to reach resources via peer networks.

Distributing the same service using anycast can eliminate or mitigate some of these problems, while introducing others. Access to a local anycast node via a peer might allow service to be maintained to an ISP with a transit failure, for example, but might also make the service vulnerable to rapid changes in the global routing system, which results in packets from a single client switching nodes, with corresponding loss of server-side state. At layer-9, anycast deployment of service might increase costs in server management, data center rental, shipping, and service monitoring, but it might also dramatically reduce Internet access charges by shifting the content closer to the consumer. As with most real-life decisions, everything is a little grey, and one size does not fit all.

Go West, Young Man

So, suppose you're the administrator of a service on the Internet. Your technical staff have decided that anycast could make their lives easier, or perhaps the pointy-haired guy on the ninth floor heard on the golf course that anycast is new and good and wants to know when it will be rolled out so he can enjoy his own puffery the next time he's struggling to maintain par on the eighth hole. What to do?

First, there's some guidance that was produced in the IETF by a group of contributors who have real experience in running anycast services. That the text of RFC 4786 [4] made it through the slings and arrows of outrageous run-on threads and appeals through the IETF process ought to count for something, in my opinion (although as a co-author my opinion is certainly biased).

Second, run a trial. No amount of theorizing can compete with real-world experience. If you want to know whether a Web server hosting images can be safely distributed around a particular network, try it out and see what happens. Find some poor victim of the slashdot effect and offer to host her page on your server, and watch your logs. Grep your `netstat -an` and look for stalled TCP sessions that might indicate a problem.

Third, think about what problems anycast could introduce, and consider ways to minimize the impact on the service or to provide a fall-back to allow the problems to be worked around. If your service involves HTTP, consider using a redirect on the anycast-distributed server that directs clients at a non-anycast URL at a specific node. Similar options exist with some streaming media servers. If you can make the transaction between clients and the anycast service as brief as possible, you might insulate against periodic routing instability that would be more likely to interrupt longer sessions.

Fourth, consider that there are some combinations of service, protocol, and network topology that will never be good environments for anycast to work. Anycast is no magic wand; to paraphrase the WOPR [5], sometimes the only way to win is not to play.

REFERENCES

[1] J. Abley, "A Software Approach to Distributing Request for DNS Service Using GNU Zebra, ISC BIND9 and FreeBSD," ISC-TN-2004-1, March 2004: http://www.isc.org/pubs/tn/isc-tn-2004-1.html.

[2] USENIX Annual Technical Conference (USENIX '04) report, *;login:*, October 2004, page 52.

[3] J. Abley, "Hierarchical Anycast for Global Service Distribution," ISC-TN-2003-1, March 2003: http://www.isc.org/pubs/tn/isc-tn-2003-1.html.

[4] J. Abley and K. Lindqvist, "Operation of Anycast Services," RFC 4786, December 2006.

[5] "War Operation Plan Response": http://en.wikipedia.org/wiki/WarGames.

ALVA COUCH

From x=1 to (setf x 1): what does configuration management mean?

A SYSTEM ADMINISTRATION RESEARCHER CONSIDERS LESSONS LEARNED FROM LISA '07, INCLUDING THE RELATIONSHIP BETWEEN CONFIGURATION MANAGEMENT AND AUTONOMIC COMPUTING

Alva Couch is an Associate Professor of Computer Science at Tufts University, where he and his students study the theory and practice of network and system administration. He served as Program Chair of LISA '02 and was a recipient of the 2003 SAGE Outstanding Achievement Award for contributions to the theory of system administration. He currently serves as Secretary of the USENIX Board of Directors.

couch@cs.tufts.edu

THE CONFIGURATION MANAGEMENT workshop this year at LISA brushed against autonomic configuration management, but as usual "there were no takers." The lessons of autonomic control in network management (also called "self-managing systems") seemed far removed from practice, "something to think about 10 years from now." Meanwhile, many talks throughout the conference (including the keynote, a guru session, and several technical papers) discussed automatic management mechanisms, although some speakers stopped short of calling these "self-managing" or "autonomic." Autonomics were almost a theme. But, in my opinion, these speakers made few converts. I stopped to think about why this is true, and I think I have a simple explanation. It's all about meaning.

The meaning crisis that system administrators face is very similar to the crisis of meaning that plagued the programming languages community in the past: There is a difference in semantics between doing things autonomically and doing things via traditional configuration management. "Semantics" refers to "what things mean." The difference is so small, and yet so profound, that the community is not fully aware of it. But it places so crippling a wall between autonomics and traditional configuration management that it is worthy of comment in itself.

Operational and Axiomatic Semantics

In programming languages, there is a "semantic wall" between statically typed languages such as C and dynamically typed languages such as LISP. The difference between these languages seems small but is actually profound. The meaning of a C program is easily defined in terms of the operations of the base machine. This is called operational semantics. By contrast, the interactions between a LISP interpreter and the base machine are not useful to understand. Instead, one expresses the meaning of statements via axiomatic semantics: a mathematical description of the observable behavior resulting from executing statements, without reference to the underlying way in which statements are actually implemented.

To understand this subtlety, consider the difference between the semantics of the C statement x=1 and the LISP statement (setf x 1). For x=1 there is an empowering operational (also called "bottom-up") semantic model that "there is a cell named X into which the value 0x00000001 is written." The operational semantics of (setf x 1), however, are not particularly empowering. There is a symbol named x that is created in a symbol table (indexed by name), and there is a numeric atom containing the value 1, and those are associated via the property "symbol-value" of the symbol x. At a deeper level, index trees become involved. But those facts about the LISP version of x are not important and not empowering except to people developing LISP. The axiomatic ("top-down") equivalent for the meaning of this statement is that "after (setf x 1), the symbol x refers to the atom 1." The details of implementation are stripped, and only the valuable functional behavior is left.

The Semantic Wall of Configuration Management

We now face a similar semantic wall between systems that exhibit "autonomic" behaviors and systems that "automate" configuration management. The latter utilize operational semantics (like x=1), whereas the former utilize axiomatic semantics (like (setf x 1)). This difference may seem unimportant, but it is central enough to cripple the discipline.

Current configuration management tools such as BCFG2, Puppet, and Cfengine utilize an operational semantic model similar to that of x=1 in C. The "meaning" of each tool's input is "what it does to the configurations of machines." Regardless of how data is specified, its final destination in a specific configuration file or files is what it "means." For example, regardless of the way in which one specifies an Internet service, one knows that it must end up as an entry in /etc/xinetd.conf or a file in /etc/xinetd.d; its "meaning" is defined in terms of that final positioning within the configuration of the machine.

By contrast, autonomic systems are configured via axiomatic semantics; the parameters specified have no direct relationship to the actual contents of files on a machine, nor is the understanding of that correspondence important or empowering, because the relationship between the parameter and the realization of that parameter (in terms of the behavior that it engenders or encourages) is too complex to be useful. For example, a specification that "the Web server must have a response time less than 2 seconds for each request" has little to do with the actual identity of the Web server or how that result might be achieved. In a very deep sense, that information is not useful in understanding the objective.

To utilize autonomics effectively, we need to progress from a semantic model in which x=1 is defined operationally to a semantic model in which (setf x 1) is defined axiomatically. This was a big step in programming languages and is an equally daunting step in configuration management. But, as I will explain, not only do current tools not contribute to that progress, they actually work actively against it, by reinforcing practices that entrench us needlessly in operational semantics and distance us from the potential for axiomatic meaning.

Abstraction and Meaning

Current approaches to configuration management, such as Cfengine, BCFG2, and Puppet, attempt to close the gap via what some authors call "raising the level of abstraction" at which one specifies configuration. However, simply raising the level of abstraction cannot scale the semantic wall between operation and behavior. One hard lesson of programming language semantics is that it is not just necessary to "abstract" upward from the machine; one must also create a model of behavior (in an axiomatic sense) with which one can reason at a high level, and with simpler semantic properties than the full operational model. Simply raising the level of abstraction does not automatically create any model other than the existing operational model of "bits on disk." Without an empowering semantic model, it is no easier to reason about a high-level description based upon operational semantics than it is to reason about a low-level description of the same thing.

Authors of configuration management tools frequently wonder why the level of adoption of their tools is so low. The answer, I think, lies in this issue of semantics. The tools do not "make things easier to understand"; they make things that remain difficult to understand easier to construct. No matter how skillfully one learns to use the tool, one is committed to an operational semantic model, in which one must still understand what "bits on disk" mean in order to understand what a tool does. The tool thus represents "something extra to learn" rather than managing "something that one can afford to forget."

There is no doubt that current tools save much work and raise the maturity level of a site but, alas, they fail to make the result easier to understand. It is thus not surprising that less experienced administrators with much left to learn about "bits on disk" shy away from having to learn even more than before. If configuration management represents "something else to learn" rather than "something easier to master," it is no surprise that use of configuration management tools finishes dead last in priority among inexperienced system administrators. If tools are to become attractive, they must represent "less to learn" rather than "more to learn."

Modeling Behavior

A successful model of configuration semantics would allow one to avoid irrelevant detail and concentrate on important details. The gulf between "automated" and "autonomic" configuration management is so great, however (like the gulf between C and the logic-programming language Prolog), that some intermediate semantics (e.g., those in LISP) might help. If, as well, this intermediate semantics is straightforward enough to be empowering, then we have a semantic layer we can use to bridge between "automatic" and "autonomic" models. The current semantics has the character of C's `x=1`, whereas this intermediate semantics might have the character of LISP's `(setf x 1)`.

Consider, for example, the intermediate semantics of file service. The "bottom-up" semantic model of this is that what one writes into configuration files leads to some fixed binding between each client (embodied as a machine) and some file server. The "top-down" model of file service can be expressed in a much simpler way. There is some "service" (that potentially can move from server to server) and some "pool of clients" (that must share the same service), but there is no binding between that service and a particular server, because that would be irrelevant to specifying the goal of providing that service. Instead, there is an expectation of service behavior that is missing from the bottom-up semantic model. That behavioral objective is that whatever file a user writes to the service is persistent across any kind of network event or contingency, and it can be recovered later by reading it back from the service via the same pathname. The way that this behavioral objective is met is not central to reasoning about the requirement. The objective is not a property of a specific machine, but of the management process itself; if the server changes, the file still (hopefully) persists, as much by the actions of human administrators (in recovering it from backups) as by the actions of software.

By contrast, the bottom-up model of file service is limiting in simple but profound ways. Saying "server X should provide Network File System (NFS) service to client Y" (or even "some server in a pool P should provide NFS to client Y") is similar to saying X=1. There are implicit limiting assumptions about how this might be done. In particular, we have implicitly decided in the former statement that NFS is the objective rather than a means toward an objective. A file service is semantically very much like (setf x 1), in which, by some unspecified method, two operations to which we refer as "write" (such as executing (setf x 1) in LISP) and "read" (analogous to referencing x after the setf) have consistent behaviors. This model is simple, but NFS is relatively complex, and there are many ways of assuring this kind of behavior other than by using NFS.

An axiomatic model of file service thus differs drastically from the operational model. The entities are not machines, but users, and the axiomatic formulation is that, for each user, writing content to a path results in that content being available henceforth via that path. For simplicity, we might notate this "behavioral axiom" as:

User –(Path:Content)-> filesystem

to mean that for an entity that is a "User," interactions with the entity "filesystem" comprise associating a "Path" with "Content" and being able to retrieve that content via that path. "User," "Path," and "Content" are types that refer to sets of potential entities, whereas "filesystem" is an entity. The arrow represents a dominance relationship, in which any entity of type "User" is dominant in creating content; "filesystem" is subservient in recording and preserving that relationship.

Modeling Services

One advantage of such a model is that many details that are purely implementation drop out of the model. The most important facet of a DHCP relationship between server and client is that the server specifies the address of the client:

DHCP –(MAC:IPaddress)-> Client

whereas the client is accessible through that address. There are many ways of assuring the latter, but one of the more common is "dynamic DNS," in which:

    DHCP –(Name:IPaddress,IPaddress:Name)->DNS

This means that DHCP specifies the name-IP mapping to DNS in accordance with its data on active clients. DNS returns this to the clients via:

    DNS –(Name:IPaddress,IPaddress:Name)->Client

This means that a client asking for an "IPaddress" for a "Name" or a "Name" for an "IPaddress" gets the one that DHCP specified originally. The "Name" to "MAC" mapping is specified by an administrator, e.g.:

    Administrator->(MAC:Name)->DHCP

These are all dominance relationships very much like the one that describes file service.

This level of detail is independent of irrelevant detail, such as how this mapping is accomplished. Caching, timeouts, and formats of mappings are (at this level) irrelevant details. The important details include dominance relationships and behavioral predictions, including that the address assigned by DHCP is indeed the address by which the host can be located via DNS.

The beauty of this scheme is that we describe “how things should work” but not “how this behavior is assured.” The former is empowering; the latter is more or less irrelevant if our tools understand the former. But current tools do not understand the former; neither can they assure this behavior without a lot of help from human beings.

Promises, Promises . . .

The astute reader will realize that this notation is very similar to that of promise theory, and the even more astute reader will realize that promise theory does not include a globally valid semantic model. Promises are a concept introduced by Mark Burgess to provide a simple framework for modeling interactions between agents during configuration management. A promise is a declaration of behavioral intent, whose semantic interpretation is up to the individual agent receiving each promise. A promise between agents assumes as little as possible about behavior, while at the same time being as clear as possible about the intention of the promise. The “type” of a promise is a starting point for the agent’s determination of the promise’s “meaning,” which is an emergent property of the promise, tempered by local observation by the receiver of its validity or lack thereof.

My notation, by contrast, globally defines expected interactions and their results. Promises enable local interactions, whereas the notation here attempts to describe overarching intent. Thus my semantics may look as though it describes promises but, because it describes intent as a global invariant, it is not like promises at all. Promise theory is one level up from my model in complexity, in not assuming that agents can be trusted to cooperate.

Coming to Closure . . .

In like manner, anything that implements the semantics of my notation is a closure, in the sense that it exhibits semantic predictability based upon an exterior description of behavior. This is how “closure” is defined.

It is well documented that building a closure is difficult and requires changes in the way we think about and notate a problem, but so is building a LISP interpreter in C, and we managed to do that. Most of the difficulties inherent in both tasks (building a closure or building a LISP interpreter) lies in letting go of lower-level details and scaling the semantic wall without looking back or down.

This is what we currently cannot bring ourselves to do.

And, because this is exactly what autonomic tools do, we are setting ourselves up for a rude awakening in which our tools and practices lag far behind the state of the art.

Science, Engineering, or Sociology?

We, our tools, and our practices are faced with a semantic wall. On one side of the wall lie operational semantics. On the other side lie axiomatic semantics. We have two choices: Scale that wall ourselves or let someone else scale it for us. If we sit still and let others do the climbing, that climbing will be done by systems engineers who understand little of the human part of system administration. If we instead take an active role, higher-level semantics can evolve in accordance with our human needs as system administrators, in addition to the needs of our organizations.

And the way I think we can take an active role may be somewhat surprising. One can take a role in this revolution even if one uses no tools and does everything by hand!

The Power of Commonality

It is easy to forget that the widely accepted Common LISP standard was preceded by a plethora of relatively uncommon LISPs. There are a million different ways to create a LISP language that conforms to the LISP axioms for behavior. But there aren't currently a million LISP implementations to match these interpretations, because high-level semantics become more useful if there is one unique way to describe their meanings in operational terms. Even though the operational semantics of LISP are not particularly easy for the novice to grasp, these same semantics give the expert a strong and universally shared semantic model that aids in performance tuning and in debugging of the interpreter itself. If one person fixes a bug in this common model, everyone using the model benefits from the fix.

This is a hard fact for the typical system administrator to swallow. We pride ourselves in molding systems in our own images. We locate files where we can find them, and we structure documentation according to personal taste. This all comes with "being the gods of the machine," as one system administrator put it. Our tools, molded in our images, support and enforce the view that customization and molding systems to our own understandings is a necessary part of management.

It is not.

There are, in my mind, roughly three levels of maturity for a system administrator:

- Managing a host
- Managing a network
- Managing business process and lifecycle

As one matures, one gradually understands and adopts practices with increasingly long-term benefits of a broader view. But even at the highest level of maturity in this model, one is not done. There remains another level of understanding and achievement:

- Managing the profession

Managing the profession entails doing things as part of one's practice that benefit all system administrators, and not just the administrators at one's own site.

It would have been easy to allow LISP to "fragment" into many languages, at no cost to the individual programmer. There would have been, though, a cost to the profession if there were 100 LISPs. It would have limited sharing and would have stifled development.

But this is exactly the juncture where we sit with configuration management now. There are a million ways to assure behavior, and everyone has a different way. Our tools support and encourage this divergence. It is like having a million different LISPs with the same axiomatic semantics and different implementations, for no particularly good reason!

In other words, a semantic model is not enough to take system administration to the next level. That model must also be shared and common, and it must refer to and be implemented via shared base semantics.

To raise the level of modeling, it is necessary to do the following:

- Avoid incidental complexity and incidental variation.
- Seek shared standards.
- Evolve a common semantic base from those standards.
- Incorporate best practices in that base.

The end product of this process is a set of shared standards that form a common semantic base that tools can implement and support.

What does this mean to you? It is really simple: If something hurts the profession, stop doing it. One example of a hurtful practice is our arbitrarily differing ways for assuring behavior. We place our personal need to remember details over the professional need for standards and consistency.

- There are millions of ideas for where packages and files should be located in a running system. Let's all choose one and stick with it!
- There are millions of ways to configure services, all of which accomplish the same thing. Let's choose one of these.
- Life is much simpler if, for example, we choose as a profession to run each service on an independent virtual server.
- Let us endeavor to leave every system in a state any other professional can understand.

Let us utilize our tools not for divergence, but for convergence to a common standard for providing and maintaining services that is so strong in semantics that we can forget the underlying details and "close the boxes." Let us support each other in protecting those standards against deviations that foster personal rather than professional objectives. If there is exactly one "best way" to provide a service, then we can all use that way, and the "institutional memory" of the profession as a whole becomes smaller and more manageable.

Will this ever happen? That is not a question of science, but one of sociology. Tool builders build their careers (and livelihoods) by encouraging adoption of "their personal views" on semantic intent. Meanwhile, the tools we have available for configuration management are still at the x=1 stage. One can throw abstraction at a problem—without semantics—and the intrinsic difficulty of the problem does not change. Only when we can define function based upon the abstraction, rather than upon its realization, can we move beyond abstraction to a workable semantics for configuration in which the internals of the configuration process become unimportant, as they rightly deserve to be.

This will be hard work, socially and technically, but the end product will be a profession whose common mission is to make all networks sing.

ERIC LANGHEINRICH

http:BL: taking DNSBL beyond SMTP

Eric Langheinrich is CTO and Co-founder of Unspam Technologies, Inc., and an expert in the field of detection and identification of malicious network activity. Eric and his team at Unspam pioneered the Project Honey Pot.

eric@eric.unspam.com

FOR MANY YEARS, EMAIL RECIPIENTS have benefited from the use of various Domain Name Blacklists (DNSBLs) in the fight against spam. Through efficient DNS lookups, mail servers can check individual connecting clients against various blacklists. Major DNSBLs include SpamHaus, SORBS, SURBL, and MAPS. These DNSBLs provide mail servers with the ability to decide how client requests are handled from hosts based on individual blacklist criteria. Hosts are able to decide to block requests, allow requests, or perform extra spam filtering scrutiny on messages from hosts based on results from blacklist lookups.

Mail servers, however, are not the only network resource that needs to be protected from malicious machines. Web servers face a constant assault from malicious Web robots that are harvesting email addresses, looking for exploits, and posting comment spam. At the most basic level, these malicious robots rob a site of significant bandwidth. More importantly, by allowing these robots to troll your site you open yourself to the possibility of future attacks via spam or Web-based vulnerabilities.

Project Honey Pot's (www.projecthoneypot.org) new http:BL service is similar to a traditional mail server DNSBL, but it is designed for Web traffic rather than mail traffic. The data provided through the service empowers Web site administrators, for the first time, to choose what traffic is allowed onto their sites. By stopping malicious robots before they can access a Web site, the http:BL service is designed to save bandwidth, reduce online threats, and decrease the volume of spam sent to the gateway by preventing spammers from getting email addresses in the first place.

Each day, thousands of robots, crawlers, and spiders troll the Web. Web site administrators have few resources available to tell whether a visitor to a site is good or malicious. Project Honey Pot was created as an open community to provide this information to Web site administrators, enabling them to make informed decisions on whom to allow onto their sites.

Project Honey Pot is a distributed network of decoy Web pages that Web site administrators can include on their sites to gather information about robots, crawlers, and spiders. The project collates data on harvesters, spammers, dictionary attackers, and comment spammers and makes this data available to its members to help them protect their Web sites and inboxes.

Web site administrators who want to participate in providing data to Project Honey Pot do so by installing a script on their site. Web site administrators include hidden links on their existing pages to the honeypot script. The links are designed to be hidden from human visitors but followed by robots. The honeypot script, when accessed, produces a Web page. Hidden on the page are trap elements, including unique email addresses and Web forms. If information is sent to these trap elements, then it is recorded by Project Honey Pot and included in the http:BL. Scripts are published open source and are currently available for PHP, Perl, ASP, Ruby, ColdFusion, and SAP NetWeaver.

Currently, Project Honey Pot has tens of thousands of installed honeypots and members in over 114 countries spanning every continent but Antarctica. Members can also participate in the project by "donating" MX records from their domains to the project. Donated MXs extend the network, allowing Project Honey Pot to track spam servers and dictionary attackers. Donated domains allow Project Honey Pot to generate a virtually unlimited number of spam trap email addresses that are difficult to detect. Together, these resources help gather information on malicious Web robots.

The http:BL service makes this data available to any member of Project Honey Pot in an easy and efficient way. To use http:BL, a host need simply perform a DNS lookup of a Web visitor's reverse IP address against one of the http:BL DNS zones. Then http:BL's DNS system will return a value that indicates the status of the visitor. Visitors may be identified as search engines, suspicious, harvesters, comment spammers, or a combination thereof. The response to the DNS query indicates what type of visitor is accessing the Web site, the threat level of the visitor, and how long it has been since the visiting IP was last seen on the Project Honey Pot network.

Each user of http:BL is required to register with Project Honey Pot. Each user of http:BL must also request an Access Key to make use of the service. All Access Keys are 12 characters in length, are lowercase, and contain only alpha characters (no numbers).

All queries must include your Access Key followed by the IP address you are seeking information about (in reverse-octet format) followed by the List-Specific Domain you are querying. Imagine, for example, you are querying for information about the IP address 127.9.1.2 and your Access Key is abcdefghijkl, then the format of your query should be constructed as follows:

```
abcdefghijkl.2.1.9.127.dnsbl.httpbl.org
[Access Key]  [Octet-Reversed IP]  [List-Specific Domain]
```

Two important things to note about the IP address in the query: First, the IP address is of the visitor to your Web site about which you are seeking information; second, the IP address must be in reverse-octet format. This means that if the IP address 127.9.1.2 visits your Web site and you want to ask http:BL for information about it, you must first reverse the IP address to be formatted as 2.1.9.127.

Note that if you reverse the order of the octets (the numbers separated by the periods) you do not reverse the IP address entirely. For example, if you were querying the IP address 10.98.76.54, the following are examples of correct and incorrect examples of reverse-octet format:

```
Query: 10.98.76.54
Right: 54.76.98.10
Wrong: 45.67.89.01
```

Three scenarios exist for responses from the http:BL service. The cases include (1) not listed, (2) listed, and (3) known search engine. A majority of IP addresses do not appear in http:BL's records. If the IP queried does not appear, http:BL will return a nonresult {NXDOMAIN}. A query for a listed entry or search engine will receive a reply from the DNS server in IPv4 format with three of the four octets containing data to provide information about the visitor. The intention is for this to allow flexibility in how the Web site administrator treats the visitor rather than a simple black-and-white response (e.g., the administrator may want to treat known harvesters differently from known comment spammers, by blocking the former from seeing email addresses while blocking the later from POSTing to forms).

Responses for listed entries will have one of two predefined formats depending on whether the entry is for a known search engine or for a malicious bot. The fourth octet represents the type of visitor. Defined types include "search engine," "suspicious," "harvester," and "comment spammer." Because a visitor may belong to multiple types (e.g., a harvester who is also a comment spammer) this octet is represented as a bitset with an aggregate value from 0 to 255. A chart outlining the different types is shown in Table 1. This value is useful because it allows you to treat different types of robots in different ways.

Value	Meaning
0	Search Engine
1	Suspicious
2	Harvester
4	Comment Spammer
8	[Reserved for Future Use]
16	[Reserved for Future Use]
32	[Reserved for Future Use]
64	[Reserved for Future Use]
128	[Reserved for Future Use]

TABLE 1

Because the fourth octet is a bitset, visitors who have been identified as falling into multiple categories may be represented. See Table 2 for an explanation of the current possible values.

IPs are labeled as "suspicious" if they engage in behavior that is consistent with a malicious robot but malicious behavior has not yet been observed. For example, on average it takes a harvester nearly a week from when it finds an email address to when it sends the first spam message to that address. In the meantime, the as-of-yet-unidentified harvester's IP address is seen hitting a number of honeypots, not obeying rules such as those set forth by robots.txt, and otherwise behaving suspiciously. In this case, the IP may be listed as suspicious.

The third octet represents a threat score for the queried IP. This score is assigned internally by Project Honey Pot based on a number of factors, such as the number of honeypots the IP has been seen visiting and the damage done

Value	Meaning
0	Search Engine (0)
1	Suspicious (1)
2	Harvester (2)
3	Suspicious & Harvester (1+2)
4	Comment Spammer (4)
5	Suspicious & Comment Spammer (1+4)
6	Harvester & Comment Spammer (2+4)
7	Suspicious & Harvester & Comment Spammer (1+2+4)
>7	[Reserved for Future Use]

TABLE 2

during those visits (email addresses harvested or forms posted to). The score ranges from 0 to 255, where 255 is extremely threatening and 0 indicates that no threat score has been assigned.

Project Honey Pot assigns threat scores to IP addresses observed on the Project Honey Pot network as part of the http:BL service. Threat scores are a rough guide to determine the threat that a particular IP address may pose and therefore should be treated as a rough measure. Although threat scores range from 0 to 255, they follow a logarithmic scale, which makes it extremely unlikely that a threat score over 200 will ever be returned.

Different threats calculate threat scores slightly differently. For example, a threat score of 25 for a harvester is not necessarily as threatening as a threat score of 25 for a comment spammer. A harvester's threat score is determined based on its reach (the number of honeypots it has hit), its damage (the number of email messages that have resulted from its harvests), its activity (the frequency of visits over a period of time), and other factors.

The second octet represents the number of days since the last activity was observed by the IP on the Project Honey Pot network. This value ranges from 0 to 255 days. This octet is useful in helping you assess how stale the information provided by http:BL is and, therefore, the extent to which you should rely on it.

The first octet is always 127 and is predefined to not have a specified meaning related to the particular visitor.

The following is an example of a hypothetical query and hypothetical response, which will be referenced throughout the rest of this section:

```
Query: abcdefghijkl.2.1.9.127.dnsbl.httpbl.org
Response: 127.3.5.1
```

The response means the visitor has exhibited suspicious behavior on the Project Honey Pot network, has a threat score of 5, and was last seen by the project's network 3 days ago.

Search engines represent a special case. Known search engines will always return a value of zero as the last octet. It is not possible for a search engine to be both a search engine and some kind of malicious bot. Search engines found to be harvesting or comment spamming will cease to be listed as search engines.

In the case of a known search engine indicated by the fourth octet being 0, the third octet becomes a serial number identifier for the specific search engine. The second octet is reserved for future use.

With the launch of the http:BL service, Project Honey Pot released the module mod_httpbl Apache. The mod_httpbl module provides an efficient mechanism for Web site administrators to take advantage of Project Honey Pot's http:BL service. The mod_httpbl module provides for server-level decision making based on http:BL data.

Rules are defined within the httpd.conf (Apache configuration) file and are indicated by the HTTPBLRBLReqHandler directive. Rules are structured in the form of [A] : [B] – [C] : [D] – [E] : [F] Action String where:

[A]—A bitmask [0–255] of the HTTP methods (in decimal representation). For example:
1—GET
2—POST
4—HEAD
8—PUT
[B]—The lower bound for DNSBL value octet 2
[C]—The upper bound for DNSBL value octet 2
[D]—The lower bound for DNSBL value octet 3
[E]—The upper bound for DNSBL value octet 3
[F]—A bitmask [0–255] of the offending type (in decimal representation) that should match
ACTION_STRING—Action to take when rule is matched. Currently supported options include "allow," "deny," and "allow-xlate-emails." For example, consider the following Action String:

```
# Serve all search engines, but replace email address text and links in HTML content.
HTTPBLRBLReqHandler 255:0-255:0-255:0 allow-xlate-emails
# Deny known comment spammers the ability to POST.
HTTPBLRBLReqHandler 2:0-255:0-255:4 deny
# Serve all harvesters, but replace email address text and links in HTML content.
HTTPBLRBLReqHandler 255:0-255:0-255:2 allow-xlate-emails
# Deny known exploiters the ability to request via HTTP method HEAD
HTTPBLRBLReqHandler 4:0-255:0-255:8 deny
# Deny any requests originating from IPs known to Project Honey Pot to be suspicious
or offensive.
HTTPBLRBLReqHandler 255:0-255:0-255:255 deny
```

The Apache module allows for granular rule sets to be defined based on an HTTP method bitset at the directory, virtual host, and server levels. The mod_httpbl rules extend basic "allow or deny" functionality to include redirects to virtual Honey Pot pages and the rewriting of email links and email address text to customized values automatically based on the http:BL query result values. For example, consider this rule set:

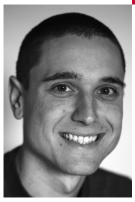
```
<VirtualHost>
HTTPRBLReqHandler (criteria_aaa)
<Directory ~ ^/dir1/>
HTTPRBLReqHandler (criteria_bbb)
HTTPRBLReqHandler (criteria_ccc)
</Directory>
HTTPRBLReqHandler (criteria_ddd)
<Directory ~ ^/dir1/images/>
HTTPRBLReqHandler (criteria_eee)
HTTPRBLReqHandler (criteria_fff)
</Directory>
HTTPRBLReqHandler (criteria_ggg)
</VirtualHost>
```

Members of the Project Honey Pot community have also provided http:BL implementations back to the community, including implementations for Drupal, phpBB, WordPress, and OddMuse. Additionally, members have posted sample code for several scripting languages in the http:BL development bulletin boards.

In a continuing effort to reduce threatening traffic, Project Honey Pot, in conjunction with its members, will continue to identify sources of malicious traffic and provide open platform tools to Web site administrators to help safeguard systems, save bandwidth, reduce online threats, and decrease the volume of spam sent to the gateway by preventing spammers from getting email addresses in the first place.

JUSTIN SAMUEL, JEREMY PLICHTA, AND JUSTIN CAPPOS

centralized package management using Stork



Justin Samuel is an undergraduate student at the University of Arizona with an interest in security. In addition to research, he teaches a secure Web application development course.

jsamuel@cs.arizona.edu

Jeremy Plichta is a senior majoring in computer science at the University of Arizona. He plans on pursuing a career as a software engineer after graduating.

jplichta@cs.arizona.edu

Justin Cappos is a Ph.D. student at the University of Arizona. His research interests revolve around building large distributed systems.

justin@cs.arizona.edu

MANAGING THE SOFTWARE INSTALLED on multiple systems can be one of the duller aspects of system administration. One has to deal with varied sets of packages, each replicated on numerous machines, and bring up new systems, with the complication of those that are almost like the others, but not quite. In many cases, great amounts of time could be saved and more than a few mistakes avoided by using tools specifically created to make this job easier.

A Better Way

Picture an admin sitting down to upgrade a certain package on 40 boxes, install new software on 100 others, and remove an unused package from every box on the network. Rather than undergo hours of tedium, the admin starts up her Stork management tool and five minutes later she's done. She knows that her systems are hard at work applying the changes she's specified. What about the systems that are currently offline? No problem. They'll apply the changes the next time they come online. And newly purchased systems? They'll be automatically updated to have the correct configuration.

Stork is the name for a collection of package management tools. Among the features it provides are:

- Central management of packages that should be installed on a distributed set of computers.
- The ability to organize systems into logical groups for ease of management.
- Increased speed and reliability of file transfers by utilizing a variety of efficient and redundant protocols.
- Secure architecture utilizing public-key cryptography for digital signatures.
- Low-upkeep repository not requiring a central repository administrator.
- Space, network bandwidth, and CPU savings in some virtual machine environments.
- Fast and efficient update awareness through the use of a publish/subscribe system for update notification.

In this article we'll focus on how to use the Stork tools for centralized management.

How Stork Works

There are three fundamental components of the Stork architecture:

- Client tools: These are package dependency resolution tools similar to yum and apt. In addition to dependency resolution, they also follow instructions from signed configuration files.
- Management tools: The management tools are the utilities administrators use to create the signed configuration files that act as the instructions to the client tools. The management tools are generally used by an administrator on his or her own computer.
- Repository: A repository is a Web server where configuration files created by the management tools are stored so that the client tools can access them.

Administrators can use one of the management tools to specify which packages should be installed on a node. The management tools create the necessary configuration files, digitally sign them with the user's private key, and can even upload them to a repository.

The client tools, running on each machine, determine which packages to install, upgrade, or remove based upon the contents of the configuration files the administrator created. If the actions require downloading package files from the repository, the client tools also verify that the package files to be installed are themselves trusted according to the configuration files.

Currently, the client and management tools run on UNIX-like operating systems, with active usage including Fedora, Gentoo, and Arch Linux. The package formats supported at this time are RPMs and tarballs, with deb file support in development. The client tools can wrap around any existing package management utilities that are installed on a system, so support can be added for any other package formats a user may desire.

Signatures and Trust

By having the client tools ensure that every configuration file downloaded from the repository is signed by a trusted administrator, the client tools can be certain that no configuration files have been tampered with. This allows unrelated organizations to safely share the use of a single repository: The client tools do not trust files because they are signed by a repository key, but rather because they are signed by an actual administrator's key.

As verifying a digital signature requires knowing in advance the public keys whose signatures should be trusted, this begs the question: How do the client tools know which public keys those are? There are two main approaches to making sure the client tools know which administrator keys to trust. These are not mutually exclusive.

The first approach is for an administrator to directly configure the client tools to trust specific keys. For example, trusted keys can be configured at the time the client tools are installed on each system. After that initial setup, new keys can be securely distributed via packages installed using Stork, for example.

The other approach that can be used is to integrate the client tools with an existing method an organization has to distribute or to answer queries for such keys. For example, PlanetLab [1] makes available the ability for a system to fetch the keys that belong to that system's administrators. Since a module in the client tools makes use of this secure PlanetLab API that provides access to these keys, the client tools can be used on PlanetLab without the need to manually distribute trusted keys to each system.

The client tools can be used in environments with multiple administrators. To accommodate this, the client tools will ensure that, for any given configuration file, they are using the most recent version of the file in the repository that has a valid signature of a trusted administrator. Thus, when another administrator uploads a new version of the file in which the groups are defined, the client tools will use the newer one.

Organizing Systems into Groups

An important optimization for managing systems is a group. A group is simply a logical association of multiple systems defined by the administrator of those systems. The fact that groups are purely a logical organization is very important: The systems in a group do not need to be connected in any way and a single system can be a member of multiple groups. Groups exist for the sole purpose of simplifying the job of the administrator, who may want multiple systems to have some of the same packages installed.

Stork has three types of groups: simple, composite, and query result.

Simple groups allow users to manage a collection of systems as if they were a single system. One can declare that machines 1, 2, and 3 are part of group G and can then simply say that package X should be installed on group G. This would result in each of machines 1, 2, and 3 installing package X. There is no limit to how many systems can be in a group.

Composite groups are created by performing an operation on existing groups. The supported operations for combined groups are UNION, INTERSECT, COMPLEMENT, and DIFFERENCE. Composite groups do not have systems directly added. Instead, the systems in the group are chosen by the operation and the membership of the other groups. For example, combined group H may be defined as the UNION of group I and group J and will contain all of the systems appearing in either group I or group J. Groups I and J can be any types of groups, including composite groups.

Query result groups are computed based on some property of the network. PlanetLab is a major user base of Stork and therefore the management tools provide support for building groups from the result of CoMon [2] queries. For example, a group to refer to all nodes with more than 1 GB of free disk space could be created using the CoMon query `select='gbfree > 1'`. Queries are evaluated at the time of group creation. Administrators who wish to have query result groups that automatically update can reevaluate the query result groups periodically via a cron job.

Management Tools

There are two main management tools for administering systems that are running the client tools: a graphical tool (a.k.a. the GUI) and a command-line tool called storkutil. These tools make it easy for administrators to create the signed configuration files that need to be uploaded to the repository. Both tools are cross-platform and require that python and openssl be installed. Let's first take a look at using the GUI to manage systems.

The information that the GUI asks for when it is first started is a username and password that will be used to authenticate with the repository, as well as the location of a private key (with optional password). The private key is used to sign configuration files. The corresponding public key is what the client tools will use to verify that the signatures on downloaded configuration files are legitimate.

After the GUI verifies the repository login information and validity of the private key, the user can begin configuring the groups and desired package installation actions. The GUI retrieves the latest configuration files from the repository and displays any groups and package installation actions that were previously defined. To add a new group, click on the "add group" button and give the group a name. Next, proceed to either add nodes to form a simple group, make this a combined group by defining it as a union or intersection of other groups, or make it a dynamic group by setting a network property such as a CoMon query.

After the new group is created, it important to define package management actions for the group. That is, define packages that should be installed, upgraded, or removed for all of the nodes of the group. To install or upgrade a package, either specify the name of the package (e.g., "gnupg") and let the client tools, when they run on the systems, attempt to find a gnupg package file that is trusted or provide a gnupg package file that is stored locally. When providing a package file that is stored locally, the GUI will take care of adding trust of this specific package file to the configuration files in addition to uploading the package file to the repository so that it is available.

If any changes are made in the GUI, an icon shows that the local state is out of sync with the repository. One can then sync with the repository, which means that the new configuration files and any newly added package files will be uploaded to the repository.

The GUI is the most convenient tool for day-to-day management of systems using the client tools. However, some situations require command-line tools that can perform these tasks. All of the same functionality for configuration file modification and generation that is done by the GUI can be done using the command-line tool storkutil.py.

Here's an example using storkutil.py to add a new group with the name EXAMPLE_NODES with one system, nodeA.example.com, in the group:

```
storkutil.py pacgroups include EXAMPLE_NODES nodeA.example.com
```

Then, to add more systems to the same group:

```
storkutil.py pacgroups include EXAMPLE_NODES nodeB.example.com \
nodeC.example.com nodeD.example.com nodeE.example.com
```

Finally, to say that all of the systems in the new group should install gnupg:

```
storkutil.py pacpackages group EXAMPLE_NODES install gnupg
```

The command-line tool storkutil.py will have taken care of generating the configuration files as well as signing them with the private key. The one thing that the GUI does that storkutil.py doesn't is upload the files to the repository for us. Uploading the files to the repository is something that can be scripted, if desired (for example, using curl).

After the configuration files are generated and uploaded to the repository using either the GUI or the command-line tools, the client tools running on any of the nodes will retrieve these newest files and take any actions necessary based on them.

Where Stork Can Help

Stork's design makes it quite flexible in terms of the roles it can perform for an administrator. However, its primary focus, and that for which it is optimized, is centralized management of many systems.

It is common for administrators to have a base configuration for all of their systems. Various systems may then have specialized software needs depending upon the additional job requirements of the people using those systems or the services they provide. For example, everybody in the finance department may need a database program that others in the organization do not. Further, all of the secretaries and executives may require a specific calendar management program. Stork makes it easy to manage the software installed for different groups of users. You can even allow a system to be in multiple groups, thus allowing the finance department's secretary to have both the database and the calendar management software installed.

Stork's system of secure file transfer, fast and efficient propagation of changes, and ability to use tarballs as packages make it very useful not only as a package management system but also as a configuration management system. For example, suppose an administrator wants to install a set of custom scripts that cron will run periodically on all of his systems to monitor disk usage. The administrator can package these scripts as a tarball and use the management tools to install the tarball on all machines. The fact that Stork manages tarballs in a similar way to RPM packages gives the administrator considerable control and flexibility without the overhead that can be involved in using traditional package formats to meet these needs. The tarball packages can be easily checked for installation, removed cleanly, or even used to satisfy dependencies.

Incorporating package file security decisions made by other users is simple with Stork's user trust system. This trust system allows an administrator to include, in real time, another user's repository-published list of untrusted packages into his or her own list. This can be very useful in many organizations with a separate IT department that focuses on security. Although this department may publish lists of packages that should not be installed because of known security problems, traditionally it can be hard for system administrators to keep up with the latest changes to such published lists. However, by being able to automatically include the other department's package trust decisions, no further work is needed to stay current with the list of packages to avoid, which can consist of packages marked as untrustworthy by name, version, or hash. With this setup, the system administrators can know that Stork will automatically prevent these untrustworthy packages from being installed.

Other Solutions

There are other tools available for performing centralized management, but none are intended to work the same way as Stork. These other systems are generally configuration management systems that can be used for some degree of package management.

One method used for configuration management is remote command execution. The tools that provide the ability to remotely execute commands on a set of nodes all utilize multiple secure shell connections that originate from the administrator's own computer. Of these, many are oriented toward specific networks, such as PlanetLab. One such system is pssh [3], which provides parallel versions of the openssh tools, including versions of ssh and scp. Using pssh, an administrator could execute a package management command on all of their active systems simultaneously. Major disadvantages with this approach include having to manually repeat the process for new systems brought online or for systems that were previously down and have been brought back online; limited groups functionality (done by utilizing files that contain lists of hosts), but having no functionality similar to Stork's composite groups; and an administrator having to make potentially hundreds or thousands of shell connections directly from the administrator's own computer in order to install, upgrade, or remove packages.

Other solutions that provide similar functionality, all using this same approach, include:

- Plush [4], which includes a graphical interface (see page 32).
- pShell [5], a command-line-only and PlanetLab-specific tool.
- PlMan [6], which includes a graphical interface and the ability to find PlanetLab nodes by CoMon queries.

Although package management on large numbers of systems would be easier using these tools compared with having to manually access each system to do so, none of them focuses on providing efficient and easy-to-use centralized package management. Using them for this purpose would be far from ideal, as none of them handles common cases such as nodes that are down at the time a command is issued or automation of setting up new nodes that come online.

A more robust way to perform configuration management is to use a powerful configuration management system such as Cfengine [7]. Configuration management systems usually involve a central server that provides configuration information to all nodes, where this configuration information is often in a format or language specific to the system. These systems provide considerable general-purpose system administration functionality, such as ensuring that certain files on all nodes are the same, configuring backups, and instructing nodes to perform basic package management actions such as installing a specific package.

Configuration management systems, however, often lack the specialized functionality for secure, centralized package management that Stork provides. For example, Stork can be used securely even when an administrator does not run his or her own repository but instead uses a shared repository. As no actions are taken by the Stork client tools unless the signatures of configuration files are verified as having been created by an administrator, the security of the nodes using the repository does not depend on the security of the repository itself.

Stork is not intended to be a configuration management system, just as tools such as Cfengine are not intended to provide the package management functionality that Stork does. In general, Stork provides a high degree of flexibility and security for centralized management, with the focus being on package management. The package management functionality of Stork, which includes very easy-to-use and lightweight packaging by means of tarball packages, allows it to function as a configuration management system in many cases.

Conclusion

For administrators responsible for multiple systems, centralized package management such as that provided by Stork can save time as well as prevent frustration and mistakes. An administrator need only define once the package management actions the systems should perform. Using Stork alleviates much of the burden of administering large numbers of systems. For more information or to download Stork, please visit the Stork Web site at http://www.cs.arizona.edu/stork.

REFERENCES

[1] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating System Support for Planetary-Scale Network Services," *Proceedings of NSDI '04* (USENIX Association, 2004).

[2] K. Park and V.S. Pai, "CoMon: A Mostly-Scalable Monitoring System for PlanetLab," *ACM Operating Systems Review*, 40(1) (January 2006): http://www.cs.princeton.edu/nsg/papers/comon_osr_06/comon.pdf.

[3] pssh: http://www.theether.org/pssh/.

[4] J. Albrecht, C. Tuttle, A.C. Snoeren, and A. Vahdat, "PlanetLab Application Management Using Plush," *ACM Operating Systems Review*, 40(1) (January 2006): http://www.cs.williams.edu/~jeannie/papers/plush-osr06.pdf.

[5] "pShell: An Interactive Shell for Managing Planetlab Slices": http://www.cs.mcgill.ca/~anrl/projects/pShell/.

[6] "Planetary Scale Control Plane": http://www.cs.washington.edu/research/networking/cplane/.

[7] M. Burgess, "Cfengine: A Site Configuration Engine," *USENIX Computing Systems*, 8(3), 1995.

JEANNIE ALBRECHT, RYAN BRAUD, DARREN DAO, NIKOLAY TOPILSKI, CHRISTOPHER TUTTLE, ALEX C. SNOEREN, AND AMIN VAHDAT

managing distributed applications with Plush

Jeannie Albrecht is an Assistant Professor of Computer Science at Williams College in Williamstown, Massachusetts. She received her Ph.D. in Computer Science from the University of California, San Diego, in June 2007 under the supervision of Amin Vahdat and Alex C. Snoeren.

jeannie@cs.williams.edu

Ryan Braud is a Ph.D. student at the University of California, San Diego, where he works under the direction of Amin Vahdat in the Systems and Networking Research Group. His interests include high-performance file distribution protocols and tools for building and debugging distributed systems, among others.

rbraud@cs.ucsd.edu

Darren Dao is a graduate student at the University of California, San Diego, where he works under the direction of Amin Vahdat in the Systems and Networking Research Group. He received his B.S. in Computer Science from the University of California, San Diego, in 2006.

hdao@ucsd.edu

Nikolay Topilski is pursuing his M.S. in Computer Science at the University of California, San Diego, where he also received his B.S. He works with Amin Vahdat, and his area of concentration is distributed network applications.

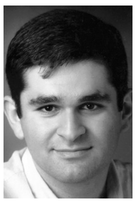
ntopilski@gmail.com

Christopher Tuttle is a Software Engineer at Google in Mountain View, California. He received his M.S. in Computer Science from the University of California, San Diego, in December 2005 under the supervision of Alex C. Snoeren.

ctuttle@google.com

Alex C. Snoeren is an Assistant Professor in the Computer Science and Engineering Department at the University of California, San Diego, where he is a member of the Systems and Networking Research Group. His research interests include operating systems, distributed computing, and mobile and wide-area networking.

snoeren@cs.ucsd.edu



Amin Vahdat is a Professor in the Department of Computer Science and Engineering and the Director of the Center for Networked Systems at the University of California, San Diego. Before joining UCSD in January 2004, he was on the faculty at Duke University from 1999 to 2003.

vahdat@cs.ucsd.edu

BUILDING AND MANAGING DISTRIButed applications is difficult. In addition to the usual challenges associated with software development, distributed applications are often designed to run on computers spread across the Internet, and therefore they have to be robust to highly variable32 network conditions and failures that are inevitable in wide-area networked environments. As a result, developers spend a significant portion of their time deploying and debugging distributed applications on remote machines spread around the world. Plush aims to ease this management burden for a broad range of distributed applications in a variety of execution environments.

No one can deny the success of the Internet. With the growing popularity of pocket-sized network-capable devices such as the iPhone, the Internet has become an integral part of our society, working its way into every aspect of our lives. As the number of Internet users continues to increase, the user demand for Internet-based services, including banking Web sites, news Web sites, and search engines, also increases. In response to this growth, many of these services require more computing power to achieve acceptable levels of performance. In short, companies need more than a single computer—or even a single room full of computers—to satisfy the computing needs of their customers. Thus, more and more services are turning to *distributed applications*, which spread their workload among a distributed set of computers, to help meet the user demand.

Distributed applications have the potential to drastically improve the scalability, fault tolerance, and reliability achieved by an Internet-based service. However, building distributed applications also introduces many new challenges to software developers. When using a distributed set of resources, developers need mechanisms for locating and configuring remote computers and for detecting and recovering from the failures that are inherent in distributed environments. In response to these challenges, many developers write complex scripts to help automate the tasks of connecting to resources, installing the needed software, starting the execution, and monitoring performance. Most of these scripts are customized to work for a specific application in a specific execution environment, and thus they are not easily extended to help other developers with similar goals.

We want to find a solution to this problem by providing a general-purpose distributed application management system that simplifies these management tasks for a broad range of applications in a variety of execution environments. Ultimately, we hope to eliminate the need for customized management scripts for deploying, running, and monitoring distributed applications in any wide-area networked environment.

Plush to the Rescue

Our solution to the problem is a system called Plush. Plush is a distributed application management infrastructure that leverages the insight that there are many similarities in the common tasks provided by customized management scripts. In particular, the first step is typically to locate and configure resources capable of hosting the application. After the resources are configured with the required software, the execution is started. Upon the completion of an execution, the scripts perform "cleanup" actions to ensure that the resources are left in a usable state for future executions. Rather than reinventing the same functionality repeatedly for each application and each execution environment, Plush automates these tasks and allows developers to define their application- and environment-specific details separately, making it easy to run applications in different distributed environments without rewriting or recreating customized scripts.

By eliminating the need for customized management scripts, Plush allows a wider range of developers to deploy and manage distributed applications running on hundreds of machines worldwide. Plush provides software developers with a user-friendly graphical user interface (GUI) called Nebula so that even novice developers can experiment with a distributed set of resources for hosting their applications. For more experienced developers, Plush also provides a command-line interface for managing distributed applications. Finally, for developers who wish to interact with the functionality of Plush from within a program or script, Plush exports an XML-RPC interface.

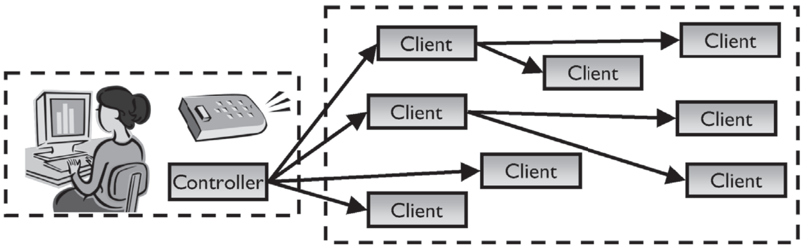


FIGURE 1: THE PLUSH CONTROLLER CONNECTS TO THE PLUSH CLIENTS RUNNING ON THE REMOTE RESOURCES.

Aside from the user interfaces, the architecture of Plush consists of two main components: the controller and the client. In most usage scenarios, the Plush controller runs locally and responds to input received from the software developer. The clients run on all remote resources involved in an application. In order to achieve scalability in Plush, the controller and clients build a communication tree for exchanging messages (Fig. 1). After establishing this tree, the controller communicates with the clients throughout the duration of an application's execution, both by sending instructions and by exchanging application management and status information. The Plush user interfaces interact directly with the controller, giving the developer a way to "remotely control" the resources hosting the application in a user-friendly way.

Plush in Action

To gain a better understanding of how Plush works, in this section we describe the tasks that Plush performs to manage a typical distributed application. These tasks are illustrated in Figure 2. More detailed information about the design and implementation of Plush can be found in our paper [1].

STEP 0: DESCRIBE THE APPLICATION

Before Plush can manage an application, the developer must provide the Plush controller with a description of the application and the desired resources for hosting the application. Typically, this is accomplished by creating a Plush application specification. When using Nebula (the Plush GUI), software developers create their application specification using a set of application "building blocks" that can be combined in an arbitrary fashion to define the custom control flow for their executions. There are separate blocks for describing resources and processes, so that developers are free to deploy applications on different resources without redefining any aspect of their execution. A resource in Plush is any computing device capable of connecting to the network and hosting an application. Developers use arrows connecting blocks to indicate the order in which various processes run within an execution. The right side of Figure 3 illustrates a sample application specification that uses the Plush building blocks. For command-line users, an XML file defines the application specification, which is loaded at the Plush prompt at startup.

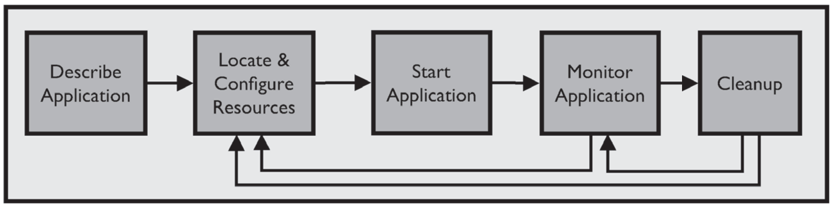


FIGURE 2: TASKS COMPLETED DURING A TYPICAL PLUSH MANAGED EXECUTION.

STEP 1: LOCATE AND CONFIGURE DESIRED RESOURCES

Once a developer creates an application specification, Plush has all of the information it needs to manage and configure a specific distributed application using a particular set of resources. From the developers' perspective, their job is done! At this point they can sit back and let Plush assume control of the application. After parsing the application specification provided by the developer, the Plush controller begins to locate and configure the desired resources. Depending on the target execution environment, this may involve using an external resource discovery service such as SWORD [7] to find resources with specific characteristics or creating new virtual machines for hosting the application with Shirako [5] or Usher [6]. Once the controller creates or locates the resources, the controller installs the Plush client software and then initiates a connection to each client. The clients' first task is to install the required software on the remote resources. Each client separately obtains the needed software packages and runs any necessary installation commands and then sends a message to the controller indicating that software installation is complete.

STEP 2: START THE APPLICATION

After the controller determines that a sufficient number of resources have been successfully configured, the controller instructs each client to start the application's execution. The clients continue to inform the controller about application status changes throughout the duration of the execution. Some distributed applications operate in phases, where each resource involved in the application must compete a specific phase of execution before any resource proceeds on to the next phase. These applications typically require some form of distributed synchronization to ensure that the phases execute correctly across all resources. Plush provides support for a range of synchronization requirements in distributed applications [2]. In order to provide this synchronization, the controller maintains a list of each client's status at all times. The controller then determines when it is safe to allow an application to move on to the next phase of execution, and it instructs the clients accordingly. Therefore, not only does Plush manage the initial starting of the application, but it also ensures that multi-phased applications start each phase of execution at the correct time.

STEP 3: MONITOR THE APPLICATION'S EXECUTION

Detecting and recovering from failures is one of the most challenging aspects of running an application on resources spread around the world. To accomplish this in Plush, the clients running remotely monitor the status of the application and resources. If a client detects a problem, ranging from insufficient disk space to unexpected program termination, the client sends a message to the controller describing the failure. The controller then decides how to recover from the problem. Plush provides built-in mechanisms for recovering from many common failures, and in most cases, Plush is able to detect and recover from errors before the developer is even aware that a problem occurred. Some failures may require finding new resources for hosting the application, whereas others may only require restarting a failed process on a single resource. For more elaborate application-specific recovery, developers can use the Plush XML-RPC interface to implement their own failure-recovery routines and then register to receive callbacks from the controller when Plush detects failures. The GUI also lets users visualize their execution with color-coded dots on a map of the world (shown on the left side of Fig. 3), allowing them to easily monitor the status of their application by simply watching the dots change colors. Thus developers who use Plush no longer need to spend a significant portion of their time writing monitoring scripts and babysitting executions running on a distributed set of machines in order to keep their applications running.

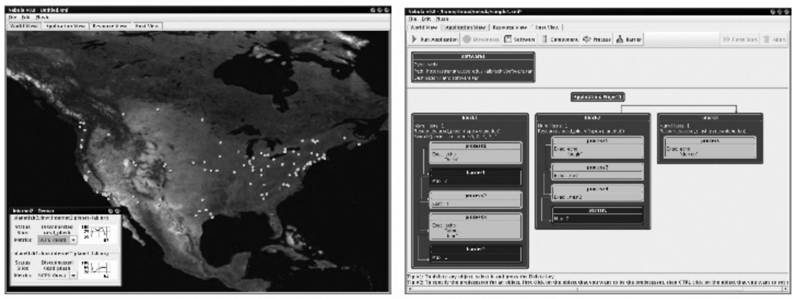


FIGURE 3: LEFT: NEBULA SHOWING PLANETLAB [4] RESOURCES RUNNING AN APPLICATION. RIGHT: AN APPLICATION SPECIFICATION BUILT USING PLUSH APPLICATION BUILDING BLOCKS.

STEP 4: CLEAN UP RESOURCES

The final task that Plush completes is to clean up the resources that host the application so that they are left in a usable state for future applications or future phases of execution. The cleanup procedure ensures that all processes exit cleanly, removing any unnecessary files and returning the state of each resource to what it was before the execution began. In general, this procedure may run at any time during the application's execution, although in most applications it is typically only run between phases or at the completion of the execution. When the controller receives messages from all clients indicating that the execution has ended (or receives input from the developer indicating that the execution should be aborted), the controller instructs the clients to kill any remaining processes associated with the application. After killing all processes, the clients also remove any unnecessary files that were created as a result of the execution. Once all clients complete the cleanup actions, the controller instructs the clients either to continue with the next phase of execution or to disconnect from the Plush communication tree and stop the client process.

Performance Evaluation

To demonstrate how well Plush recovers from failures in a wide-area networked environment, Figure 4 evaluates Plush's ability to detect host failures and subsequently to find and configure replacement resources for SWORD running across PlanetLab. SWORD is a wide-area resource discovery service that requires each host to download and install a 38-MB software package before starting the execution. PlanetLab is a distributed execution environment consisting of 800+ resources spread across 40+ countries. In this experiment, Plush starts SWORD on 100 randomly selected PlanetLab machines, including some machines behind DSL network links. After 1250 seconds, we manually kill SWORD on 20 of the initial 100 machines to simulate host failures. The Plush clients independently notify the controller of the failures, and the controller locates and configures replacement resources for the ones that failed. The SWORD service is fully restored across 100 machines 1000 seconds later.

Using Plush to manage this application allowed us to avoid writing a custom script that probed for and recovered from host failures. Particularly for long-running services such as SWORD, developers need automated mechanisms for monitoring the behavior of the execution and coping with problems that arise. It is unrealistic to expect the developer of a service to constantly monitor its performance, but at the same time, a service must quickly and automatically recover from failures since other developers may rely on the functions that it provides. When using Plush, clients running on the PlanetLab resources monitor the service's performance at all times and automatically recover from failures. More details about this experiment are discussed in the paper by Albrecht et al. [1].

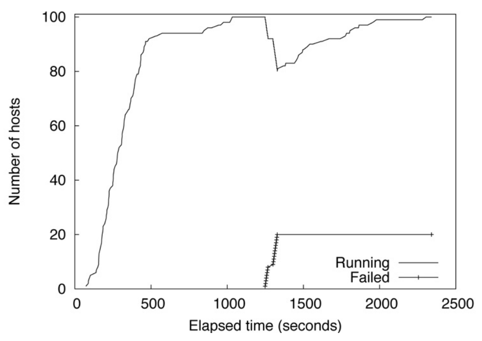


FIGURE 4: PLUSH INITIALLY STARTS SWORD ACROSS 100 PLANETLAB RESOURCES. AFTER 1250 SECONDS, 20 OF THESE RESOURCES FAIL. PLUSH AUTOMATICALLY DETECTS AND REPLACES THE FAILED HOSTS AND RESTORES THE APPLICATION.

How Do I Use Plush?

Plush is an open-source, publicly available software package that can be obtained from the Plush Web page [8]. Plush is implemented in C++, and it runs on most UNIX-based platforms. It depends on several C++ libraries, including those provided by xmlrpc-c, curl, xml2, zlib, math, openssl, readline, curses, boost, and pthreads. In addition, the command-line user interface requires packages for lex and yacc. (We typically use flex and bison.) If you intend to use Plush on PlanetLab, the controller uses several simple Perl scripts for interacting with the PlanetLab Central database. The only requirement related to network connectivity is that the Plush controller must be able to SSH to all remote resources.

Nebula is also publicly available. Nebula is implemented in Java and runs on any platform that supports Java, including most UNIX-based platforms, Windows, and Mac OS X, among others. Nebula communicates with the Plush controller using the XML-RPC programmatic interface. XML-RPC is implemented in Nebula using the Apache XML-RPC client and server packages. One additional benefit of using Nebula is that because it communicates with the Plush controller solely via XML-RPC, it is not necessary to run Nebula and the Plush controller on the same machine. If Nebula and Plush run on separate machines, after starting Nebula locally, developers have the option, using the Nebula preference menu, of specifying a Plush controller process running remotely.

Plush is currently in daily use worldwide. We have used Plush to successfully manage a variety of distributed applications, ranging from long-running services to short-lived, multi-phased computations. These applications were run in several different resource environments, including PlanetLab, ModelNet [9], and clusters of Xen [3] virtual machines. Although user feedback thus far has been largely positive, our goal is to make Nebula and Plush as user-friendly as possible, so we welcome all comments, suggestions, and feedback. If you would like further information, please visit our Web site (http://plush.cs.williams.edu), and feel free to contact any of the authors if you have additional questions.

REFERENCES

[1] J. Albrecht, R. Braud, D. Dao, N. Topilski, C. Tuttle, A.C. Snoeren, and A. Vahdat, "Remote Control: Distributed Application Configuration, Management, and Visualization with Plush," *Proceedings of the USENIX Large Installation System Administration Conference (LISA)*, 2007.

[2] J. Albrecht, C. Tuttle, A.C. Snoeren, and A. Vahdat, "Loose Synchronization for Large-Scale Networked Systems," *Proceedings of the USENIX Annual Technical Conference (USENIX)*, 2006.

[3] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the Art of Virtualization," *Proceedings of the ACM Symposium on Operating System Principles (SOSP)*, 2003.

[4] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating Systems Support for Planetary-Scale Network Services," *Proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[5] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. G. Yocum,

"Sharing Networked Resources with Brokered Leases," *Proceedings of the USENIX Annual Technical Conference (USENIX)*, 2006.

[6] M. McNett, D. Gupta, A. Vahdat, and G.M. Voelker, "Usher: An Extensible Framework for Managing Clusters of Virtual Machines," *Proceedings of the USENIX Large Installation System Administration Conference (LISA)*, 2007.

[7] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat, "Design and Implementation Tradeoffs for Wide-Area Resource Discovery," *Proceedings of the IEEE Symposium on High Performance Distributed Computing (HPDC)*, 2005.

[8] Plush Web page: http://plush.cs.williams.edu.

[9] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić J. Chase, and D. Becker, "Scalability and Accuracy in a Large-Scale Network Emulator," *Proceedings of the ACM/USENIX Symposium on Operating System Design and Implementation (OSDI)*, 2002.

OCTAVE ORGERON

an introduction to logical domains

PART 3: ADVANCED NETWORKING AND STORAGE

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

unixconsole@yahoo.com

IN THE OCTOBER 2007 ISSUE OF *;login:*, I walked you through the installation and configuration of the Logical Domain (LDom) technology. In this article, I'll talk about advanced topics concerning configuration networking and storage. The discussion here will give you a deeper understanding of the two most challenging resources to administer with LDoms.

Is It a Guest Domain?

When one is remotely logged in, it is difficult to determine whether the environment is in a guest domain. The Solaris operating system will behave and function like a normal standalone server. The most obvious place to see the difference is in the device tree:

```
ldom1:~ $ cat /etc/path_to_inst
#
# Caution! This file contains critical kernel state
#
"/iscsi" 0 "iscsi"
"/pseudo" 0 "pseudo"
"/scsi_vhci" 0 "scsi_vhci"
"/options" 0 "options"
"/virtual-devices@100" 0 "vnex"
"/virtual-devices@100/channel-devices@200" 0 "cnex"
"/virtual-devices@100/channel-devices@200/network@0" 0 "vnet"
"/virtual-devices@100/channel-devices@200/network@1" 1 "vnet"
"/virtual-devices@100/channel-devices@200/disk@0" 0 "vdc"
"/virtual-devices@100/channel-devices@200/disk@1" 1 "vdc"
"/virtual-devices@100/console@1" 0 "qcn"
"/virtual-devices@100/ncp@6" 0 "ncp"
"/virtual-devices@100/random-number-generator@e" 0 "n2rng"
"/virtual-devices@100/n2cp@7" 0 "n2cp"
```

Notice how short the device tree is and that many of the devices are under the "/virtual-devices@100" nexus. All of the networking and storage devices are virtualized, a clear indicator that the system you are logged into is a guest domain. Beyond these hardware subtleties, it is difficult for a user or developer to tell the difference. However, from an administrative point of view, networking and storage are critical aspects of any environment.

Advanced Networking

Networking with LDoms can be complicated, depending on the goals you wish to accomplish. As such, it is important to have a clear understanding of what is possible where networking is concerned.

Virtual switches, or VSWs, are virtual devices that provide the functions of a basic layer 2 network switch and packet demultiplexer. The VSW enables network communications between guest domains on the same VSW and the external network to which the VSW is connected. For internal traffic, the VSW classifies incoming packets based on the target VNET MAC address and switches the packets to the destination VNET device. For external traffic, it acts like a forwarding agent between VNET interfaces and external clients.

The MAC address of a VSW or a VNET can be automatically assigned or specified during creation. Normally, the automatic assignment of MAC addresses should not cause conflicts with other MAC addresses on your networks. However, if a conflict does arise, you can specify the MAC address. This can also be handy if you are going to use DHCP for assigning IPs or have a specific application requirement. For example:

```
primary:~ # ldm add-vsw mac-addr=8:20:4f:ab:cd:ef net-dev=e1000g6 primary-vsw6 primary
primary:~ # ldm add-vnet mac-addr=00:14:4f:f9:c6:96 ldom1-vnet2 primary-vsw6 ldom1
```

This may also be useful if you want the MAC address for a VSW to be the same as the physical network port to which it is connected:

```
primary~ # ifconfig e1000g0
e1000g0: flags=201000802<UP,BROADCAST,MULTICAST,IPv4,CoS> mtu 1500 index 2
    inet 192.168.2.12 netmask ffffff00 broadcast 192.168.2.255
    ether 0:14:4f:96:f6:72

primary:~ # ldm add-vsw mac-addr=0:14:4f:96:f6:72 net-dev=e1000g0 primary-vsw0 primary
```

By default, when a VSW is created, the primary domain is incapable of communicating directly with the guest domains on that VSW. For the primary domain to be able to communicate with the guest domains directly, it must be connected to the VSW. This can be done by either plumbing the VSW in addition to the physical network device or only plumbing the VSW. The second option consumes fewer IPs and can prevent confusion:

```
primary:~ # ifconfig e1000g0 down unplumb
primary:~ # mv /etc/hostname.e1000g0 /etc/hostname.vsw0
primary:~ # svcadm restart network/physical
primary:~ # ifconfig vsw0
vsw0: flags=201000802<UP,BROADCAST,MULTICAST,IPv4,CoS> mtu 1500 index 2
    inet 192.168.2.12 netmask ffffff00 broadcast 192.168.2.255
    ether 0:14:4f:96:f6:72
```

This plumbing enables the primary domain to communicate directly with the guest domains that are connected to the VSW without having to transverse the physical network. However, this can expose your primary domain to insecure networks. The primary domain should be protected from public access because of its importance. If the primary domain does not need to be connected to the same network your guest domains are on, the underlying physical network device or VSW can be left unplumbed.

VSWs can also be configured for private networking. Such private networks can be useful for configuring communications only among LDoms. This is accomplished by not specifying a physical network device to bind with the VSW:

```
primary:~ # ldm add-vsw primary-vsw4 primary
primary:~ # ldm list-bindings primary
...
NAME           MAC                NET-DEV      DEVICE       MODE
primary-vsw3   00:14:4f:96:f6:75  e1000g3      switch@3     prog,promisc
```

```
  primary-vsw4     00:14:4f:f8:d4:a6               switch@4 routed
...
```

Notice the differences between these VSWs. When a physical network device is specified, the VSW is enabled with layer 2 switching with the underlying hardware in programmed and promiscuous mode. If a physical network device is not specified, the VSW is enabled with layer 3 IP routing in nonpromiscuous mode.

If a physical network port were to go offline, this could potentially cause a major outage for your guest domains. One of the ways you can reduce the risks of such an outage is to configure IPMP (IP Multi-Pathing), thereby enabling multiple network paths to handle the fail-over of IPs in the event of a NIC or connection failure. IPMP also load-balances outgoing traffic. The easiest option is to configure it within the guest domain. This is accomplished by configuring at least two VNETs into a guest domain that are connected to separate VSWs on the same physical network, then configuring IPMP with IP-probe-based error detection in the guest domain:

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

ldom1:~ # cat /etc/hostname.vnet0
192.168.2.101/24 broadcast + group net1 -failover deprecated up
addif ldom1/24 broadcast + up

ldom1:~ # cat /etc/hostname.vnet1
192.168.2.102/24 broadcast + group net1 -failover deprecated up

ldom1:~ # ifconfig -a

...
vnet0: flags=209040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,
NOFAILOVER,CoS> mtu 1500 index 2
      inet 192.168.2.101 netmask ffffff00 broadcast 192.168.2.255
      groupname net1
      ether 0:14:4f:fb:49:89
vnet0:1: flags=201000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4,CoS> mtu 1500 index 2
      inet 192.168.2.100 netmask ffffff00 broadcast 192.168.2.255
vnet1:
flags=209040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,
CoS> mtu 1500 index 3
      inet 192.168.2.102 netmask ffffff00 broadcast 192.168.2.255
      groupname net1
      ether 0:14:4f:fb:f3:12
```

IP-based probing configures the interfaces to ping each other to determine a failure. This determination is required with guest domains because the link status information from the physical network devices is not propagated up through the VSWs. Although such a procedure will provide IP fail-over for your guest domain, it does consume additional IPs for each guest domain you configure with IPMP. Other methods are discussed in the LDom administrator guide [1].

Networking features such as bridging, VLAN tagging, and link aggregation are not yet supported with LDoms. However, they are being worked on in the OpenSolaris community [2].

Advanced Storage

As was demonstrated in the previous article, storage can be virtualized in interesting ways to support LDoms. The difficult part is figuring out which storage option to use. The chart in Table 1 can be used as a quick reference.

Storage Option	Jumpstart	Virtualized for Guest Domains	Non-Virtualized for Guest Domains
DASD	Yes	Yes	No
SAN	Yes	Yes	No
iSCSI	Yes	Yes	Yes
ZFS volumes	Yes*	Yes	Yes
SVM meta-devices	Yes*	Yes	Yes
Virtual disk images	Yes	Yes	No
NAS	No	No	Yes

* Currently, only works with Solaris Express Build 75 and higher.

TABLE 1: STORAGE OPTIONS

DASD, SAN, iSCSI, virtual disk images, ZFS volumes, and SVM meta-devices can be virtualized as either boot or data storage for guest domains. Let's start out and take a look at DASD:

```
primary:~ # ldm add-vdsdev /dev/dsk/c3t2d0s2 ldom4-vdsk0@primary-vds0
primary:~ # ldm add-vdsdev /dev/dsk/c3t3d0s2 ldom4-vdsk1@primary-vds0
primary:~ # ldm add-vdisk ldom4-vdsk0 ldom4-vdsk0@primary-vds0 ldom4
primary:~ # ldm add-vdisk ldom4-vdsk1 ldom4-vdsk1@primary-vds0 ldom4

ldom4:~ # format
...
        0. c0d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
          /virtual-devices@100/channel-devices@200/disk@0
        1. c0d1 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
          /virtual-devices@100/channel-devices@200/disk@1
...
```

DASD storage is simple and consumes little overhead. However, you are limited by the amount that can be installed internally or attached externally to your server. In the event of a system failure, that storage would have to be manually migrated to a standby server.

SAN storage has many benefits, including performance, reliability, and portability among servers. Solaris 10 and above include support for SAN connectivity. Although third-party drivers and FC HBAs may work on standalone servers, they may not work with LDoms. This is also true with SAN multi-pathing software. STMS (a.k.a. MPXIO) should be utilized. STMS can only be utilized in a service domain that has direct access to the FC HBAs, such as the primary domain. The SAN infrastructure and multi-pathing are completely transparent to guest domains. This situation will change in the future when FC HBAs can be virtualized using NPIV, which provides guest domains with a virtualized FC HBA instance [3]. Here is an example of virtualizing SAN storage:

```
primary:~ # ldm add-vdsdev /dev/dsk/c6t60060160B5681200B2CE5AD37981DB11d0s2 ldom5-vdsk0@primary-vds0
primary:~ # ldm add-vdsdev /dev/dsk/sc6t60060160B5681200B3CE5AD37981DB11d02 ldom5-vdsk1@primary-vds0
primary:~ # ldm add-vdisk ldom5-vdsk0 ldom5-vdsk0@primary-vds0 ldom5
primary:~ # ldm add-vdisk ldom5-vdsk1 ldom5-vdsk1@primary-vds0 ldom5

ldom5:~ # format
...
        0. c0d0 <DGC-RAID5-0219 cyl 32766 alt 2 hd 64 sec 10>
          /virtual-devices@100/channel-devices@200/disk@0
        1. c0d1 <DGC-RAID5-0219 cyl 32766 alt 2 hd 64 sec 10>
          /virtual-devices@100/channel-devices@200/disk@1
...
```

iSCSI has some interesting advantages over SAN storage owing to its low entry costs and the ubiquity of Ethernet. Combined with dedicated networking or 10Gb Ethernet, it can be a viable solution for virtualization. iSCSI targets can be virtualized for guest domain storage and they can be used directly by guest domains. However, OpenBoot does not support iSCSI, so neither standalone servers nor guest domains can boot directly from it at this point. Hopefully, this will change in the future.

For iSCSI to be utilized as boot storage for a guest domain, it must first be virtualized:

```
primary:~ # iscsiadm list target
Target: iqn.1986-03.com.sun:02:8a59994a-b4df-4968-d1e1-a63e4229bd2c
        Alias: storage/ldom3-vdsk0
        TPGT: 1
        ISID: 4000002a0000
        Connections: 1
Target: iqn.1986-03.com.sun:02:6eae2782-e453-ea54-c39b-d87bca8636de
        Alias: storage/ldom3-vdsk1
        TPGT: 1
        ISID: 4000002a0000
        Connections: 1

primary:~ # ldm add-vdsdev /dev/dsk/c2t01000003BA16E64B00002A004747D0F9d0s2 ldom3-vdsk0@primary-vds0
primary:~ # ldm add-vdsdev /dev/dsk/c2t01000003BA16E64B00002A004747D0FBd0s2 ldom3-vdsk1@primary-vds0

primary:~ # ldm add-vdisk ldom3-vdsk0 ldom3-vdsk0@primary-vds0 ldom3
primary:~ # ldm add-vdisk ldom3-vdsk1 ldom3-vdsk1@primary-vds0 ldom3

ldom3:~ # format
...
        0. c0d0 <SUN-SOLARIS-1 cyl 32766 alt 2 hd 4 sec 160>
           /virtual-devices@100/channel-devices@200/disk@0
        1. c0d1 <SUN-SOLARIS-1 cyl 32766 alt 2 hd 4 sec 160>
           /virtual-devices@100/channel-devices@200/disk@1
...
```

Guest domains can also make use of iSCSI for additional storage directly:

```
ldom1:~ # iscsiadm list target
Target: iqn.1986-03.com.sun:02:aa8c6f76-52b5-e3f3-9725-f8c3bbb18fbf
    Alias: storage/ldom1-vdsk2
    TPGT: 1
    ISID: 4000002a0000
    Connections: 1

ldom1:~ # format
...
    2. c1t01000003BA16E64B00002A004747B609d0 <SUN-SOLARIS-1 cyl 32766 alt 2 hd 4 sec 80>
       /scsi_vhci/ssd@g01000003ba16e64b00002a004747b609
...
```

ZFS and SVM can be used to create volumes or meta-devices that can be virtualized for LDoms, allowing a very granular level of control over how storage is utilized. These volumes will appear as local disks in a guest domain. However, in Solaris 10 these volumes are crippled in the sense that they are not virtualized as whole disks and are presented with only slice 0. This prevents them from being utilized for Jumpstart or for booting. Luckily, this has already been fixed in OpenSolaris as of build 75 and will be back-ported to Solaris 10 at some point in the future. Here is a demonstration of how ZFS volumes can be utilized with LDoms:

```
primary:~ # zfs create -V 10gb ldoms/ldom2-vdsk0
primary:~ # zfs create -V 10gb ldoms/ldom2-vdsk1

primary:~ # ldm add-vdsdev /dev/zvol/dsk/ldoms/ldom2-vdsk0 ldom2-vdsk0@primary-vds0
primary:~ # ldm add-vdsdev /dev/zvol/dsk/ldoms/ldom2-vdsk1 ldom2-vdsk1@primary-vds0

primary:~ # ldm add-vdisk ldom2-vdsk0 ldom2-vdsk0@primary-vds0 ldom2
primary:~ # ldm add-vdisk ldom2-vdsk1 ldom2-vdsk1@primary-vds0 ldom2

ldom2:~ # format
...
          0. c0d0 <SUN-DiskImage-10GB cyl 34950 alt 2 hd 1 sec 600>
             /virtual-devices@100/channel-devices@200/disk@0
          1. c0d1 <SUN-DiskImage-10GB cyl 34950 alt 2 hd 1 sec 600>
             /virtual-devices@100/channel-devices@200/disk@1
...
```

Virtual disk images provide a wide range of flexibility as they can be stored on DASD, SAN, iSCSI, and NAS. When combined with the use of advanced file systems, such as ZFS, LDoms can be quickly replicated and provisioned. Here is a demonstration of this type of flexibility:

```
primary :~ # ldm list-bindings primary | grep ldom1 | grep img
   primary-vds0    ldom1-vdsk0     /ldoms/local/ldom1/ldom1-vdsk0.img
                   ldom1-vdsk1     /ldoms/local/ldom1/ldom1-vdsk1.img

primary :~ # zfs list /ldoms/local/ldom1
NAME            USED    AVAIL   REFER   MOUNTPOINT
ldoms/ldom1     25.0G   72.0G   25.0G   /ldoms/local/ldom1

primary :~ # ldm list ldom1
NAME    STATE   FLAGS   CONS    VCPU    MEMORY      UTIL    UPTIME
ldom1   active  -n—–    5000    4       4G          0.2%    16h 11m
```

As we can see, ldom1 is using virtual disk images that are on a ZFS file system. We can utilize ldom1 as a template for future LDoms by logging into it to perform a sys-unconfig to remove the host configuration and bring ldom1 to a halt. Then we can take a ZFS snapshot and clone:

```
primary :~ # ldm list ldom1
NAME    STATE   FLAGS   CONS    VCPU    MEMORY      UTIL    UPTIME
ldom1   bound   ——–     5000    4       4G

primary :~ # zfs snapshot ldoms/ldom1@copy1
primary :~ # zfs clone ldoms/ldom1@copy1 ldoms/ldom6
primary :~ # zfs set mountpoint=/ldoms/local/ldom6 ldoms/ldom6

primary :~ # zfs list ldoms/ldom6
NAME            USED    AVAIL   REFER   MOUNTPOINT
ldoms/ldom6     16K     72.0G   25.0G   /ldoms/local/ldom6

primary:~ # ls /ldoms/local/ldom6
ldom1-vdsk0.img ldom1-vdsk1.img ldom1-vdsk2.img
```

The ZFS snapshot is a point in time copy that does not use any additional space. When we take a ZFS clone, only the data that is changed will consume space. Now let's rename the virtual disk images and add them to our new LDom:

```
primary :/ldoms/local/ldom6 # mv primary 1-vdsk0.img ldom6-vdsk0.img
primary :/ldoms/local/ldom6 # mv primary 1-vdsk1.img ldom6-vdsk1.img

primary :~ # ldm add-vdsdev /ldoms/local/ldom6/ldom6-vdsk0.img ldom6-vdsk0@primary-vds0
primary :~ # ldm add-vdsdev /ldoms/local/ldom6/ldom6-vdsk1.img ldom6-vdsk1@primary-vds0

primary :~ # ldm add-vdisk ldom6-vdsk0 ldom6-vdsk0@primary-vds0 ldom6
primary :~ # ldm add-vdisk ldom6-vdsk1 ldom6-vdsk1@primary-vds0 ldom6

primary :~ # ldm bind ldom6
primary :~ # ldm start ldom6
LDom ldom6 started

primary :~ # ldm list ldom6
NAME    STATE    FLAGS    CONS    VCPU    MEMORY    UTIL    UPTIME
ldom6   active   t—–      5003    4       2G        25%     25s
```

Once we connect to the console for ldom6 and boot it, we can configure it for usage by following the standard host configuration screens. Once that is completed, we can start using ldom6:

```
ldom6:~ # uname -a
SunOS ldom6 5.11 snv_75 sun4v sparc SUNW,SPARC-Enterprise-T5120
ldom6:~ # format
...
        0. c0d0 <SUN-DiskImage-10GB cyl 34950 alt 2 hd 1 sec 600>
          /virtual-devices@100/channel-devices@200/disk@0
        1. c0d1 <SUN-DiskImage-10GB cyl 34950 alt 2 hd 1 sec 600>
          /virtual-devices@100/channel-devices@200/disk@1
...

primary:~ # zfs list ldoms/ldom6
NAME            USED    AVAIL    REFER    MOUNTPOINT
ldoms/ldom6     382M    71.7G    25.0G    /ldoms/local/ldom6

primary:~ # du -sh /ldoms/local/ldom6
  25G /ldoms/local/ldom6
```

Notice that our cloned LDom is actually only using about 382 MB of space in the ZFS storage pool. Using this method can save considerable amounts of storage for each LDom. Since virtual disk images are just sparse files, they can be easily migrated among different storage types and even burned to DVD disks for portability.

NAS storage cannot be used for Jumpstart, but it can be utilized for storing virtual disk images. This enables NAS storage to consolidate virtual disk images for LDoms and make them easily accessible for a farm of LDom-capable servers. Here's how:

```
primary:~ # df -h /ldoms/nas/ldom7
Filesystem          size     used     avail    capacity  Mounted on
192.168.2.2:/export/ldoms/ldom7
                    92G      11G      82G      12%       /ldoms/nas/ldom7

primary:~ # mkfile 10g /ldoms/nas/ldom7/ldom7-vdsk0.img
primary:~ # mkfile 10g /ldoms/nas/ldom7/ldom7-vdsk1.img

primary:~ # ldm add-vdsdev /ldoms/nas/ldom7/ldom7-vdsk0.img ldom7-vdsk0@primary-vds0
primary:~ # ldm add-vdsdev /ldoms/nas/ldom7/ldom7-vdsk1.img ldom7-vdsk1@primary-vds0
primary:~ # ldm add-vdisk ldom7-vdsk0 ldom7-vdsk0@primary-vds0 ldom7
primary:~ # ldm add-vdisk ldom7-vdsk1 ldom7-vdsk1@primary-vds0 ldom7

ldom7:~ # format
```

```
...
      0. c0d0 <SUN-DiskImage-10GB cyl 34950 alt 2 hd 1 sec 600>
         /virtual-devices@100/channel-devices@200/disk@0
      1. c0d1 <SUN-DiskImage-10GB cyl 34950 alt 2 hd 1 sec 600>
         /virtual-devices@100/channel-devices@200/disk@1
...
```

Within a guest domain, NAS storage can be accessed via standard NFS and can be utilized for sharing common storage such as home directories, application binaries, and application data.

One of the things you may have noticed is that all of the storage examples shown here are configured with at least two virtual disks for each guest domain. This is because the virtualized boot disks for the guest domains are mirrored. The mirroring is handled by SVM and in the future ZFS. This increases the reliability of your guest domains and enables features such as Live Upgrade, which is used for Solaris release upgrades.

```
ldom1:~ # metastat -p
d20 -m /dev/md/rdsk/d21 /dev/md/rdsk/d22 1
d21 1 1 /dev/rdsk/c0d0s1
d22 1 1 /dev/rdsk/c0d1s1
d10 -m /dev/md/rdsk/d11 /dev/md/rdsk/d12 1
d11 1 1 /dev/rdsk/c0d0s0
d12 1 1 /dev/rdsk/c0d1s0

ldom1:~ # df -h /
Filesystem          size    used    avail   capacity  Mounted on
/dev/md/dsk/d10     7.9G    4.5G    3.3G    58%       /
ldom1:~ # swap -l

swapfile            dev     swaplo  blocks              free
/dev/md/dsk/d20     85,5        16  2097584          2097584
```

The mirroring of the virtual boot disks in a guest domain is only the beginning. SVM and ZFS can be utilized to manage any additional storage that is virtualized into a guest domain or connected through other means such as iSCSI, enabling a wide range of configurations and possibilities for managing storage.

There are some key things to keep in mind with LDoms and storage:

- Virtualized storage cannot be shared among guest domains. This can prevent certain storage applications, such as clustered file systems, from functioning.
- When storage is virtualized, low-level SCSI system calls are not implemented. This can affect certain third-party volume managers or applications that expect such low-level access to SCSI targets.
- Using SVM or ZFS in the service domain for managing guest domain storage can increase the I/O overhead. As such, at least 4 GB of memory should be allocated to your primary domain.
- Backups should be performed from your guest domains. If you have a guest domain that is using virtual disk images, it should be shut down before being backed up. This will prevent the backups of the images files from being “fuzzy.”

Summary

This article has demonstrated many of the advanced networking and storage configurations. This background should enable you to make informed decisions when configuring LDoms in conjunction with your networking and storage infrastructures. In the next article, I will demonstrate other advanced topics that center around hardware design, resource management, and manageability.

REFERENCES

[1] LDoms Administration Guide 1.0: http://docs-pdf.sun.com/819-6428-11/819-6428-11.pdf.

[2] OpenSolaris LDoms Community: http://opensolaris.org/os/community/ldoms/.

[3] OpenSolaris NPIV Project: http://www.opensolaris.org/os/project/npiv/.

ADITYA K SOOD

insecurities in designing XML signatures

Aditya K Sood, a.k.a. oknock, is an independent security researcher and founder of SecNiche Security, a security research arena. He is a regular speaker at conferences such as XCON, OWASP, and CERT-IN. His other projects include Mlabs, CERA, and TrioSec.

adi.zerok@gmail.com

XML DIGITAL SIGNATURE TECHNOLOGY is on high heels nowadays, but potential insecurities have been encountered because of insecure programming practices. This article discusses the weak spots in the coding of XML signatures and related operations. The procedural approach involves inline cryptography to combat application vulnerabilities. Stress is placed on secure coding practices.

This article encompasses the practical problems in designing XML signatures through the use of APIs. XML signatures are used to provide security to data of any kind, whether XML or binary. The confidentiality, integrity, and authenticity of the message have to be preserved when designing a SOAP request for communication. XML API functionality is very versatile but at the same time protection measures have to be included to prevent loss of data. Verification of data on the client end becomes a formidable task, owing to the persistence of errors, leading to failure of the post-verification signing process. The prerequisites will be listed and discussed from the application point of view to thwart Web-based errors in XML signing of messages.

XML Digital Signatures

The digital signing of messages has become an efficient security measure. XML signatures are being used extensively to initiate a realm of security. The implementation is done through Apache structural libraries or XML digital signature APIs. However, using digital signatures is not devoid of implementation problems. This article serves to dissect the process of implementation of XML API.

XML security is considered an intermediate process in designing Java security components. As such, the element of security is implemented before interaction with an application. The major component is XWSS, which stands for XML Web Service Security. This component functions directly with the Apache XML security provider.

To understand XWSS, let's look first at the XML security stack.

The APIs are standardized under JSR 105. The two pluggable components present are the Apache XML Security Provider and the SUN Java Cryptography Architecture (JCA) Provider. Both compo-

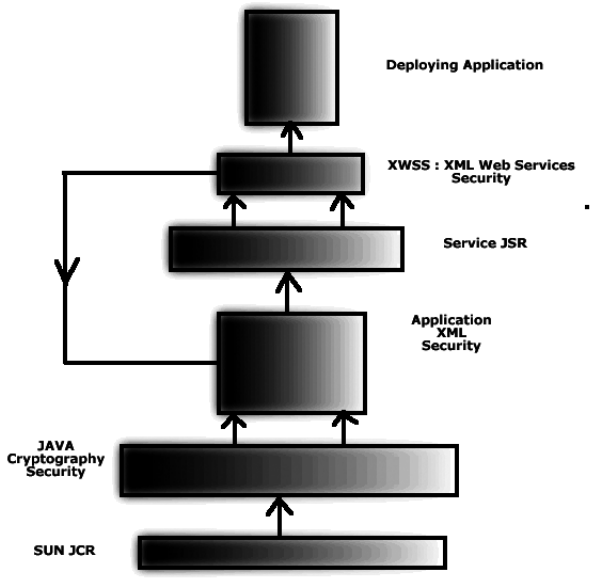


FIGURE 1: THE COMPONENTS USED IN XML SIGNING

nents interface directly with the JCA component. The JSR 105 standard is implemented through the Apache XML Security Provider, which ensures proper interaction with the XWSS component. So, overall a message is signed with an XML signature before an application receives it. The use of XML APIs is an effective method for protecting data integrity. The cryptography architecture followed is elucidated in Figure 1. The practical implementation is made possible through the design of cryptographic libraries imported during the time of execution.

Let's have a look at the XML signature packages:

- The java.xml.crypto package contains classes that are used directly in the implementation of design and generation and encryption of messages through the digital XML signature.
- The java.xml.crypto.dsig package comprises interfacial components that describe the cryptographic-related W3C specifications. It is used in signing and validation of digital signatures.
- The javax.xml.crypto.dsig.keyinfo package constitutes interfaces to various key structures that are defined in the W3C XML digital signature recommendation.
- The javax.xml.crypto.dsig.spec package contains classes for input parameters such as digests and keys.
- The javax.xml.crypto.dom and javax.xml.crypto.dsig.dom packages contain DOM-related classes.

The presentation of these various packages is initiated to trigger the JAVA Cryptography Architecture. The XML signature structures are implemented by the various interfaces provided by these crypto packages. For example, the Key-related interfaces Keyinfo, KeyName, KeyValue, and PGPData are defined on the basis of the W3C recommendations. Developers basically generate abstract factories such as XMLSignatureFactory or KeyInfoFactory based on the interfaces provided by these crypto packages. Developers also create their own URI dereferencing implementation based on the URI dereference class.

Secure coding is a precursor to secure implementation. Improper handling and implementation can marginalize the entire structure of Web applications. These implementation problems are described next.

Let’s have a look at the XML signature layout (see Fig. 2).

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="urn:envelope">
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
<SignedInfo>
<CanonicalizationMethod
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComme
nts"/>
<SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>

<Reference URI="">
<Transforms>
<Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
</Transforms>

<DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<DigestValue>uooqbWYa5VCqcJCbuymBKqm17vY=</DigestValue>
</Reference>
</SignedInfo>

<SignatureValue>KedJuTob5gtvYx9qM3k3gm7kbLBwVbEQRl26S2tmXjqNND7MRG
toew==
</SignatureValue>
<KeyInfo>
<KeyValue>
<DSAKeyValue>

<P>/KaCzo4Syrom78z3EQ5SbbB4sF7ey80etKII864WF64B81uRpH5t9jQTxeEu0Im
bzRMqzVDZkVG9xD7nN1kuFw==</P>
<Q>li7dzDacuo67Jg7mtqEm2TRuOMU=</Q>

<G>Z4Rxsnqc9E7pGknFFH2xqaryRPBaQ01khpMdLRQnG541Awtx/XPaF5Bpsy4pN
WMOHCBiNU0NogpsQW5QvnlMpA==</G>
<Y>qV38IqrWJG0V/mZQvRVi1OHw9Zj84nDC4jO8P0axi1gb6d+475yhMjSc/BrIVC
58W3ydbkK+Ri4OKbaRZlYeRA==
</Y>

</DSAKeyValue>
</KeyValue>
</KeyInfo>
</Signature>
</Envelope>
```

FIGURE 2: EXAMPLE OF AN XML SIGNATURE

The example in Figure 2 clearly depicts the implementation structure of an XML signature. It comprises a Keyinfo structure, which further incorporates the KeyValue. The full structure is placed into an envelope for transmission across the entities for secure communication. The XML signature specifications are based on the W3C recommendations and are applied directly on defined benchmarks. The benchmarks here refer to the standard specification provided by the W3C for effective structural design of XML documents and related applications. It actually provides a hierarchical implementation of XML objects. Also present is the signed info structure, which holds the desired information bearing the signature. It is implemented in a canonical form, in which a reference element is called by a URI. The value of the URI is always undertaken as a string. If the string is empty or NULL, then the root of the document is defined by that URI.

With this introduction to XML digital signatures, we can now dissect the implementation problems that cause discrepancies in communication.

Parsing Anatomy in Instantiating a Signature

The very first problem occurs in developing the instantiation of an XML digital signature. Parsing is actually undertaken by a JAXP builder library. Usually the builder library is present in a default state to be used independently. The developer can make a mistake in parsing an XML signature instance object through the predefined builder library. The proper implementation hierarchy is shown in Figure 3.The benchmark of standard XML implementation seen in Figure 3 is a basic procedure for designing an XML signature. First a signature instance object is created, then the Name space is set. Once this is done the builder library is called to parse the created object. Let’s have a look at the code:

```
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);
DocumentBuilder builder = dbf.newDocumentBuilder();
Document doc = builder.parse(new FileInputStream(argv[0]));
```

The code is stated in the hierarchy provided in Figure 3. The flaw occurs mainly in setting NameSpaceAware true and in argument-passing in parsing through file-streaming. File-streaming is a process in which file-handling functions are dynamically used based on the variance of input. Because argv[0], the value of the input parameter changes when different numbers of arguments are passed. If the proper argument is not passed, the instantiation of the XML signature goes awry, because it affects the signature stats. So parsing must be taken into account to hinder any further passing of errors in the signature. If the instantiation is not done properly, it can cause stringent errors in the application of the signature. Thus developers should be careful in accomplishing this task.

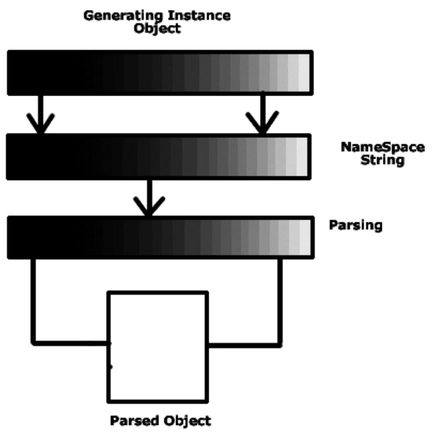


FIGURE 3: LOGICAL HIERARCHY FOR IMPLEMENTING XML SIGNATURES

Signature Specification Error Checks

Once the object is instantiated, the next step is to specify a signature, which then has to be validated. The major problem occurs when the error checks are not implemented properly, and consequently wrong elements are not filtered and get passed as such. For example, in the absence of a string check, an application error occurs whenever a null string or large string is passed. During signature specification the error checks have to be executed by the developers to ensure that security is not constrained. Overlooked errors have the potential to throw the entire application into disarray. Let's look at the code for a better view:

```
NodeList nl = doc.getElementsByTagNameNS(XMLSignature.XMLNS, "Signature");
if (nl.getLength() == 0) { throw new Exception("Cannot find Signature element");}
```

As you see, handling errors during object implementation mutes their impact.

KeySelector Problem in the Validation Context

The validation context states that the context in which an XML signature instance is validated by passing input parameters. For instance, if a developer is using a DOM (Document Object Model), the developer has to instantiate a DOM validation context instance. The problem occurs mainly in passing the reference parameters to the generated validation context. In this a Key-

Selector KeyValue and a reference to a signature element are passed. The coding flaw occurs in passing the KeySelector pair and elements. This hampers the process of validation and leads to false references.

The following code represents the implementation of KeyValue and KeySelector structures:

```
private static class KeyValueKeySelector extends KeySelector {
    public KeySelectorResult select(KeyInfo keyInfo, KeySelector.Purpose purpose, AlgorithmMethod
    method,XMLCryptoContext context) throws KeySelectorException {

if (keyInfo == null) { throw new KeySelectorException("Null KeyInfo object!"); }
    SignatureMethod sm = (SignatureMethod) method;
    List list = keyInfo.getContent();
    for (int i = 0; i < list.size(); i++) {
    XMLStructure xmlStructure = (XMLStructure) list.get(i); if (xmlStructure instanceof KeyValue) {
    PublicKey pk = null;  try { pk = ((KeyValue)xmlStructure).getPublicKey(); } catch (KeyException ke) {

throw new KeySelectorException(ke); }
// make sure algorithm is compatible with method
    if (algEquals(sm.getAlgorithm(), pk.getAlgorithm())) { return new SimpleKeySelectorResult(pk); } } }

throw new KeySelectorException("No KeyValue element found!"); }
    static boolean algEquals(String algURI, String algName) {
    if (algName.equalsIgnoreCase("DSA") && algURI.equalsIgnoreCase(SignatureMethod.DSA_SHA1)) {
    return true; } else if (algName.equalsIgnoreCase("RSA") &&
    algURI.equalsIgnoreCase(SignatureMethod.RSA_SHA1)) { return true; } else { return false;} } }
```

Actually, KeySelector tries to find a suitable key for the validation of data. The Key is stored in KeyValue. So a wrapper class is designed for applying this. Remember, to subdue the impact of KeySelector problems, KeySelector exceptions should be implemented with desired checks. The context is implemented as follows:

```
DOMValidateContext valContext = new DOMValidateContext (new KeyValueKeySelector(), nl.item(0));
```

Developers should take this into account in developing robust signatures.

Mismanagement in Assembling XML Signature Components

Once different components are designed and articulated with code, they must be assembled into a singular object. This assembly is required because the application of a signature is possible only after the completion of the centralized object (i.e., XML signature object). As stated earlier, the application calls DOM to get the handle of the required XML signature, which is possible through the XMLSignatureFactory object. Three steps must be completed prior to the implementation:

- Signing the URI of an object
- Specifying the digest method
- Transforming the enveloped layout

Whenever an application calls a specific code related to XML signatures from the server, a URI is required to complete the action. The URI describes the root of the element. If a mismatch occurs in passing arguments, information can be leveraged, because the infection vector is randomized and it can direct the execution vector in any sphere of application.

The envelope transformation causes the signature to be removed prior to the calculation of the signature value. Insecurity occurs in passing arguments. In this example, no specific object is supplied but a null string is subjected as an argument and the transformation object is set directly. This is bad programming practice in the context of signature designing. Look at this code snippet:

```
Reference ref = fac.newReference
    ("", fac.newDigestMethod(DigestMethod.SHA1, null),
    Collections.singletonList (fac.newTransform(Transform.ENVELOPED,
    (TransformParameterSpec) null)), null, null);
```

References should be applied with caution; the wrong reference points to the wrong application entity, thereby creating considerable inefficiency. The SignedInfo object should be created carefully with argument fusing. The problem here is that the first parameter in the reference is supplied as [" "], but it should be supplied with some proper argument or NULL (pointing to no memory). Reference parameters should be supplied in a correct manner with standard objects, as follows:

```
        Reference ref = fac.newReference("#object",
            fac.newDigestMethod(DigestMethod.SHA1, null));
SignedInfo si = fac.newSignedInfo (fac.newCanonicalizationMethod (CanonicalizationMethod.
INCLUSIVE_WITH_COMMENTS, (C14NMethodParameterSpec) null),
fac.newSignatureMethod(SignatureMethod.DSA_SHA1, null), Collections.singletonList(ref));
```

Once the SignedInfo object is created, key generation comes next. The keys should be handled and generated in a standard manner in the context of the specific application:

```
KeyInfoFactory kif = fac.getKeyInfoFactory(); KeyInfo object:
KeyValue kv = kif.newKeyValue(kp.getPublic());
KeyInfo ki = kif.newKeyInfo(Collections.singletonList(kv));
XMLSignature signature = fac.newXMLSignature(si, ki);
```

Strict vigilance is required for assembling XML signature components. The issues presented here cover some of the major problems related to XML signature designing.

Conclusion

Application security requires a well-planned and security-oriented coding layout to work efficiently. Dethroning insecure vectors requires secure coding practices. Application functionality can be jeopardized by the absence of even one of these factors. Protection should be applied through effective mechanisms or by adopting a security development life cycle when designing applications. Secure coding is considered to be a good proactive defense in combating application flaws.

FURTHER READING

[1] M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon, "XML-Signature Syntax and Processing," in W3C Recommendation, World Wide Web Consortium, 12 February 2002, D. Eastlake, J. Reagle, and D. Solo, editors: http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/.

[2] T. Imamura, B. Dillaway, and E. Simon, "XML Encryption Syntax and Processing," in W3C Recommendation, World Wide Web Consortium, 10 December 2002, D. Eastlake and J. Reagle, editors: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/.

[3] D. Eastlake and K. Niles, *Secure XML: The New Syntax for Signatures and Encryption* (Upper Saddle River, NJ: Pearson Education, 2002).

[4] J. Rosenberg and D. Remy, *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML Signature and XML Encryption* (Indianapolis: Sams, 2004).

[5] http://java.sun.com/security/javaone/2002/javaone02.3189-jsr105-bof.pdf.

DAVID N. BLANK-EDELMAN

practical Perl tools: why I live at the P.O.

David N. Blank-Edelman is the Director of Technology at the Northeastern University College of Computer and Information Science and the author of the O'Reilly book *Perl for System Administration*. He has spent the last 22+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and one of the LISA '06 Invited Talks co-chairs.

dnb@ccs.neu.dnb@ccs.neu.edu

WITH APOLOGIES TO THE GREAT BUT deceased Southern writer and a nod to her fan Steve Dorner, I'd like to take some time this issue to explore how Perl can help you with the business of manipulating data that resides on mail servers. Sending mail from Perl is a fairly well-known process (heck, it is even in the Perl FAQ; see perldoc perlfaq9 for more details), but the process of pulling data down from a server or moving it around on that server could use a little more explanation.

(Quick aside: I intentionally used the formulation "data on mail servers" instead of mailboxes or mail messages in that last paragraph. The AUP-breaking hacks that let you treat outsourced mail systems like Gmail as remote data stores have forever changed my view of just what is or can be stored on those servers. I'll return to the usual conventions now.)

Before we dive into the *how* of this process, I think it is reasonable for you to demand a good answer to the *why*. There are a whole host of reasons why you might want to automate mail operations like this via Perl. Some of these reasons are immediately apparent if you run the mail server in question. For example, to truly test that your mail system is working it is important to be able to check that your users can actually read their mail. I've publicly advocated that people write round-trip tests for mail systems that involve sending automatic mail to a test account that is then retrieved in a similarly automated fashion. This is far better than just a simple banner scrape to show your MTA is still listening on a socket.

If you don't run any mail servers you are still likely to encounter situations where automated mail-manipulation knowledge could be useful. For example, if your ISP does not perform spam filtering to your satisfaction, you could pull down all of the mail in your inbox, run it through whatever rigorous tests suit your fancy (no doubt involving some goat entrails) and then act on the messages before they can sully your mail reader. If that ISP can't do server-side filtering, your program could take over that job as well. The list goes on.

POP3 Goes the Weasel

Let's start someplace simple. The POP3 protocol, documented in RFC1939, offers a relatively uncomplicated way for a client to interact with a mail store. In most cases a POP3 client will:

1. Connect to a POP3 server and authenticate as a known user (known as a mailbox).
2. See if there is new mail.
3. Request the contents of the first new message and squirrel it away on the local machine.
4. Request that the server delete that message.
5. Repeat #3 and #4 for every remaining new message.
6. Signal that it is done with the connection and exit (with the server performing the actual deletion of data for the messages marked as deleted in step #4).

This set of six steps shows virtually all of the operations available in the protocol. The only two things of interest we did not mention are how "new messages" are handled and the TOP command. Let's quickly hit those two subjects in that order.

If a client always deletes all of the messages once it has downloaded them, it is trivial to determine when a message is new and requires downloading: Anything found in the mail store is by definition "new." But it isn't always advantageous to delete upon reading. The most common case where this isn't desirable is one where a user wants to have two separate POP3 clients looking at the same mailstore (e.g., your home machine and your work machine). One of them simply downloads the mail; the other will both download and delete it.

The client that doesn't delete the mail needs a way of remembering which messages it has seen before so it doesn't download them a second time. This is typically done using the POP3 UIDL command. UIDL asks the server to display a "unique-id listing" for a message or for each message on the server. This gives the client a piece of information that uniquely identifies each message on the server, which it can cache for future reference when deciding which messages to download. UIDL is officially "optional" in the RFC, but I have yet to see a modern POP3 server that didn't implement it.

I know that you are about to suffer from the DTs because you haven't seen any Perl code yet in this column, but hang on for a couple of more sentences because I want to mention one more POP3 feature I think will come in handy for you. POP3 also has an optional TOP command that allows the client to request the headers of a message followed by the first N lines of a message. This allows a client to get a peek at the contents of a message without having to download the whole thing.

Phew. With that verbiage out of the way, let's get to some code:

```
use Mail::POP3Client;

my $pop3 = new Mail::POP3Client(
    USER     => 'user',
    PASSWORD => 'secretsquirrel',
    HOST     => 'pop3.example.edu',
    USESSL   => 'true',
);

die 'Connection failed: ' . $pop3->Message() . "\n"
   if $pop3->Count() == -1;

print 'Number of messages in this mailbox: ' . $pop3->Count() . "\n\n";
print "The first message looks like this: \n" . $pop3->Retrieve(1) . "\n";

$pop3->Close();
```

This code uses the Mail::POP3Client module to connect to the POP3 server (over SSL, natch), retrieve the number of messages present, and then display the first message. There are a few POP3-oriented modules available on CPAN (the other popular one being Net::POP3), but I tend to like this one because the methods it provides mostly map directly to the commands in the protocol. If I had one quibble it would probably be that "mostly" part because I'd prefer it to offer methods I can infer from the RFC (even as an option) rather than making up its own. For example, it provides Retrieve(), also known as HeadAndBody(), whereas RFC calls that RETR.

Still, you can probably guess how you could extend this code to do more sophisticated things. Delete(message #) can be called to mark a message for deletion. Uidl() is available to you, returning an array whose contents contain the "unique-id listing" for each message or messages sought. TOP is even present in the form of a Head() method, and so on. Both the Head() and HeadAndBody() methods will return either a scalar or an array based on their calling context, so it is easy to get a mail header or message in the form desired by packages such as Mail::SpamAssassin.

IMAP and You Never Go Back

I don't want to dwell on POP3 any longer, because we have some more interesting fish to fry. The other protocol people use for interacting with their mail data is IMAP4. IMAP4 is a significantly more powerful (read: complex) protocol. Its basic model is different from that of POP3. With POP3 it is assumed that the POP3 client polls the POP3 server and downloads mail periodically. With IMAP4 a client connects to a server for the duration of the mail reading session. (Warning: There is a little hand-waving here, because of something known as disconnected mode, which we'll talk about in a sec.) With POP3 the client is expected to do all of the heavy lifting in the process. With IMAP4 the discussion between the server and the client is much richer and so the protocol has to be considerably smarter. Smarter how?

1. IMAP4 can deal with multiple mail folders and their contents (including other folders). RFC3501 says, "IMAP4rev1 includes operations for creating, deleting, and renaming mailboxes, checking for new messages, permanently removing messages, setting and clearing flags, RFC 2822 and RFC 2045 parsing, searching, and selective fetching of message attributes, texts, and portions thereof."
2. IMAP4 has support for disconnected clients. In disconnected mode a client can operate on a local cache of a mailbox even when not connected to its server. Later the client will play the changes back to the server to bring the local cache and the server's copy into sync. This is what allows you to sit on a plane without network access, deleting and filing mail, later to have those changes be propagated to the server when you get back on the Net.
3. IMAP4 has a much more granular understanding of an individual mail message. POP3 lets us grab a mail message's headers or headers plus N of the first lines of the message body. IMAP4 lets us say, "Give me the part of the message body that includes the message text but don't send me the data for the embedded attachments." It does this by grokking MIME natively.
4. Since this isn't a user-visible thing, you never hear about this last feature in POP3 vs. IMAP4 comparisons. If you watched the discussion between a POP3 client and server (actually, most client-server discussions), it would look like this: command from client, reply from server, command, reply, command, reply . . .

> With IMAP4, the client can send a slew of commands at one time and have the server send responses to any of those commands anytime in the session. The two don't have to communicate in lockstep with each other. Each command is prefixed with a unique tag that the server will repeat back at the beginning of the response for that command. This lets both sides keep track of what has been asked and what is being answered.
>
> Note that your code doesn't have to be written in a highly asynchronous manner using this capability (and in fact the examples in this column won't be), but it is good to know it exists if you do need to write high-performance IMAP4 code.

I just want to mention up front that given the complexity of the protocol, working with IMAP4 isn't always as intuitive as you'd like. Unfortunately, we don't have enough room in this column to look at all of the little squirrelly bits, so I'm going to constrain myself to very simple examples. If you start to write your own programs you *must* read the relevant RFCs (RFC2060 at a minimum, RFC2683 suggested) plus the documentation for whatever Perl module you choose to use.

For the sample code we're about to see, I'll be using my current preferred IMAP module, Mail::IMAPClient. This is the same module that forms the basis of the superb imapsync program (http://www.linux-france.org/prj/imapsync/dist/), a great tool for migrating data from one IMAP4 server to another. In addition to the vote of confidence because of imapsync, I like this module because it is mostly complete when it comes to features while still offering the ability to send raw IMAP4 commands should it become necessary. The other module that I would consider looking at is Mail::IMAPTalk by the primary developer behind Fastmail.fm. Even though it hasn't been updated in a few years, the author assures me that the current release still works well and is in active use there.

So let's dig into some IMAP4 code. As an example we'll use some code that connects to a user's mailbox, finds everything that was previously labeled as spam, and moves those messages to a SPAM folder. We'll start with connecting to the IMAP server:

```
use IO::Socket::SSL;
use Mail::IMAPClient;
my $s = IO::Socket::SSL->new(PeerAddr =>'imap.example.com',
                             PeerPort => '993',
                             Proto    => 'tcp');
die $@ unless defined $s;

my $m = Mail::IMAPClient->new(User     => 'user', Socket=>$s,
                              Password => 'topsecret');
```

This code is a little more verbose than I'd like, but I thought it was important to demonstrate how one uses an SSL connection to connect to the server. Mail::IMAPClient doesn't have SSL built in in the same way Mail::POP3Client does, so we had to construct an SSL-protected socket by hand and pass it to Mail::IMAPClient.

Once connected, the first thing one typically does is tell the server which folder to operate on. In this case we'll select the user's INBOX:

```
$m->select('INBOX');
```

Now that we have a folder selected, it's time to get to work. Let's find all of the messages in our INBOX that have the X-Spam-Flag header set to YES:

```
my @spammsgs = $m->search(qw(HEADER X-Spam-Flag YES));
die $@ if $@;
```

Now that I have a list of messages in @spammsgs, I can move each one over to the folder named SPAM:

```
foreach my $msg (@spammsgs){
  die $m->LastError unless defined $m->move('SPAM',$msg);
}
```

Once we've moved all messages we can close the mailbox and log out of the server:

```
$m->close();        # expunges currently selected folder
$m->logout;
```

There's a hidden detail in the first of these two lines of code that I feel compelled to mention. You might remember from the POP3 discussion that we talked about messages being "marked as deleted." The same tombstoning process takes place here as well. Deletes are always a two-step process in IMAP4 (flag as \Deleted and expunge messages marked with that flag). When we requested that a message be moved, the server copied the message to the new folder and marked the message in the source folder as being deleted. Ordinarily you would need to expunge() the source folder to actually remove the message, but RFC2060 says that a CLOSE operation on a folder explicitly expunges that folder, so we get away without having to do it ourselves.

I'd like to show only one more small IMAP4 example because there's still one last major topic left to cover in this column after IMAP4. I mentioned that IMAP can take apart messages (specifically, into their component MIME parts). Here's some code that demonstrates it. In the interests of saving space, I'll leave out the code from the last example that performed the initial SSL socket/connect to server and INBOX select:

```
my @digests = $m->search(qw(SUBJECT digest));

foreach my $msg (@digests) {

  my $struct = $m->get_bodystructure($msg);
  next unless defined $struct;

  # messages in a mailbox get assigned both a sequence number and
  # a unique identifier. By default Mail::IMAPClient works with UIDs
  print "Message with UID $msg (Content-type: ",$struct->bodytype,"/",
         $struct->bodysubtype,
         ") has this structure:\n\t",
         join("\n\t",$struct->parts) ,"\n\n";
}

$m->logout;
```

This code searches for all of the messages whose subject has the word "digest" in it. For each message it attempts to parse the structure of the message and print out a list of parts it finds. Here's a small snippet of output you might expect from the code:

```
Message with UID 2457 (Content-type: TEXT/PLAIN) has this structure:
    HEAD
    1

  Message with UID 29691 (Content-type: MULTIPART/MIXED) has this structure:
       1
       2
```

```
        3
        3.1
        3.1.HEAD
        3.1.1
        3.1.2
        3.2
        3.2.HEAD
        3.2.1
        3.2.2
        3.3
        3.3.HEAD
        3.3.1
        3.3.2
        4
```

If we needed to access just one of the parts of the message we can call `bodypart_string` with the message number and part number. For example:

```
print $m->bodypart_string(29691,'4');
```

prints out the footer of the message with UID 29691:

```
_______________________________________________
Perl-Win32-Database mailing list
Perl-Win32-Database@listserv.ActiveState.com
To unsubscribe: http://listserv.ActiveState.com/mailman/mysubs
```

Mail::IMAPClient uses the Parse::RecDescent module to take apart MIME messages. I find that it works most of the time but has some issues with certain messages. If you are doing a lot of MIME groveling you may find that you'll either want to call a dedicated MIME parser or look at the module Mail::IMAPTalk mentioned earlier, which has the ability to parse messages into easy Perl structures. If we used Mail::IMAPTalk to fetch the body structure of that message and turned on its spiffy parse mode, here's an excerpt of what we would see stored for the footer part of the message:

```
  3  HASH(0x85bf38)
     'Content-Description' => 'Digest Footer'
     'Content-Disposition' => HASH(0x85d9cc)
        empty hash
     'Content-ID' => undef
     'Content-Language' => undef
     'Content-MD5' => undef
     'Content-Transfer-Encoding' => '7BIT'
     'Content-Type' => HASH(0x85d0f8)
       'charset' => 'us-ascii'
     'IMAP-Partnum' => 4
     'Lines' => 4
     'MIME-Subtype' => 'plain'
     'MIME-TxtType' => 'text/plain'
     'MIME-Type' => 'text'
'Remainder' => ARRAY(0x856528)
   0  undef
'Size' => 193
```

Make All the Hairy and Scary Code Go Away

There remains one last thing to mention: If all of this gnarly POP3 and IMAP4 code has you worried, there are a few modules out there that attempt to abstract out the details necessary for dealing with mail on a POP3 or

IMAP4 server. For example, the Email::Folder family (part of the Perl Email Project, at emailproject.perl.org/wiki/Email::Folder) lets you write code like this (from the doc):

```
use Email::Folder;
use Email::FolderType::Net;

my $folder = Email::Folder->new('imaps://example.com'); # read INBOX

print $_->header('Subject') for $folder->messages;
```

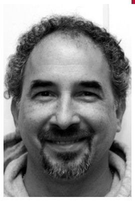
The other package worth considering is the all-singing-all-dancing MailBox. Here's what the author says: "The MailBox package is a suite of classes for accessing and managing email folders in a folder-independent manner. This package is an alternative to the Mail::Folder and MIME::* packages. It abstracts the details of messages, message storage, and message threads, while providing better performance than older mail packages. It is meant to provide an object-oriented toolset for all kinds of e-mail applications, under which Mail User Agents (MUA) and mail filtering programs [*sic*]."

MailBox is a highly engineered package with tons of functionality (which may be a good or a bad thing in your eyes). It ships with enough module tests to choke a horse (which is likely to be a good thing from your management's perspective). MailBox actually uses Mail::IMAPClient under the hood to do its IMAP4 work, but you'll never know it because it abstracts all of the IMAP4 details away for you.

With that pointer, it is time to exit. Have fun manipulating your mail data from Perl. Take care, and I'll see you next time.

PETER BAER GALVIN

Pete's all things Sun (PATS): the future of Sun



Peter Baer Galvin (http://www.galvin.info) is the Chief Technologist for Corporate Technologies, a premier systems integrator and VAR (www.cptech.com). Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is coauthor of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide.

pbg@cptech.com

HI, AND WELCOME TO A NEW COLUMN in *;login:*. Thanks are owed to USENIX for giving me the opportunity to write about "All Things Sun" in this lovely journal. I previously wrote a column for *Sys Admin* magazine (http://www.samag.com), may it rest in peace. Before that I wrote for *SunWorld,* may it also rest in peace. Hopefully this tragic cycle will be broken at *;login:*! If you've seen my previous writings, you'll know I like to cover the gamut of topics around Sun, Solaris, and sometimes more general information technology issues. Experience is the best teacher, and the best source of topics to write about, so I'll draw on my work and play experiences to try to save the reader's time (and sometimes money)—pointing out the good, the bad, and the ugly of all things Sun. Feedback is always welcome, so feel free to let me know what you think about the column and/or suggest topics to write about.

This month I thought it would be nice to cover something near and dear to those who use or are thinking about using Sun products: the future of Sun.

The Future of Sun

History is full of inflection points. And only history can tell the long-term results of those inflections. Consider Sun a few years ago. If one were to create a checklist of vendors and their advantages and features, there were many rows in which Sun lacked the all-important checkmark. Solaris 9 was a strong Sun asset, but it ran only on SPARC. Sun made only SPARC systems, and SPARC was not the best choice in many circumstances. Sun was seemingly "closed" and "proprietary." This fact was not lost on IT managers, who were consistently choosing other solutions to their post-dot-com boom infrastructure.

Fast forward to the Sun of today and ones sees that the company has certainly closed the "checkmark gap." One inflection point was the open sourcing of Solaris (and many other Sun software assets). Of course, as with most things Sun does, that move was not without controversy. Sun chose to create a new license—the CDDL—rather than use an existing one. Another inflection was the release of Solaris 10, arguably the most advanced and feature-rich operating system in history. A third was the purchase of Andy Bechtolsheim's company Kealia. Andy, when he is not being a world-class venture capitalist [1], is designing world-class x86 systems for Sun.

In November 2007 Sun inflected again. This time the area was Solaris, specifically OpenSolaris. OpenSolaris has been the tagline for the open sourcing of Solaris. And although there are several OpenSolaris distributions, there was no official "OpenSolaris distribution." Project Indiana, headed by former Debian founder Ian Murdock, has morphed into "the" OpenSolaris distribution. The OpenSolaris distribution is a brave, new Sun. For one thing, it is not 100% backward-compatible with Solaris 10. Sun prides itself on the backward compatibility of its operating systems. (In fact it was rumored to be a firing offense for a Sun engineer to break that compatibility.) But when ZFS is the root file system, and a new package system is the cornerstone of a distribution, backward compatibility apparently has to go. By combining those two aspects, OpenSolaris is able to have a liveCD format to "try before you install" (and also can be booted from a USB stick!). It also becomes more, well, Linux-like in its ease of adding new software packages from the network and having those packages manage themselves (with versioning, seamless upgrades, and dependency management).

Eggs are clearly being broken, but whether the result is an omelet or something far less tasty won't be known for a while. A preview release (with very limited functionality) of OpenSolaris (a.k.a. Project Indiana) was made available in November and is expected to be production-ready in a few months. You can find it on the OpenSolaris Web site [2]. Just what "production-ready" means is a bit of an open issue, but Sun is saying that it will support OpenSolaris for those with support contracts. A crucial question for Sun (and its customers) is how ISVs will respond to the new release. Applications make or break a distribution. Where does that leave Solaris and its next releases? Clearly those will continue for quite a while, but one version of future history has those tapering off and OpenSolaris continuing as, for all intents and purposes, *the* Solaris. This preview release is one view of the OpenSolaris future. There is considerable internal and external debate about it, so before it sees production there could be more radical changes (and less backward compatibility) or a rollback on some features to create more compatibility. The problem statement of Project Indiana is also available from the OpenSolaris Web site [3].

In the meantime, Sun engineers continue to do work on new operating system components and continue checking them in (performing a "putback") at opensolaris.org. Those projects are then cherry-picked for inclusion in Solaris releases as well as use in other distributions. There are many interesting and useful changes taking place. These are worth tracking, and trying in OpenSolaris, because they can be tested for stability and functionality well before you need to plan their use in production. Some of the most interesting projects include:

- CIFS: including CIFS as a kernel-level server
- Caiman: a redo of the Solaris installer
- Clearview, which unifies the feature set provided by network devices
- Crossbow: deep network virtualization for resource control, performance, and security
- DTrace providers, which enable DTrace to probe more languages and system aspects such as NFS V4
- ZFS on-disk encryption
- ZFS boot-ability

The full list and access points to each project's Web pages and discussion groups are online [4].

Perhaps the most major recent opensolaris.org putback was the inclusion of the "xvm" project. This project is based on the Xensource virtualization code. Once this project is ripe, Solaris will be the "Dom0," or host, operating system and Solaris will support many "DomU" guests, including Windows, Linux, BSD, Solaris x86, and pretty much anything that runs on x86 hardware. This feature will only work on x86 hardware (of course), so a new inflection will be Solaris x86 having different features from Solaris SPARC. Solaris SPARC on UltraSPARC T1 and T2 CPUs has its own hypervisor-like virtualization called LDoms (covered in the August and October 2007 issues of *;login:*). Add to that the CPU-independent Solaris Containers and there is a wealth (overabundance?) of virtualization choices for Sun, which would be a good topic for a future column.

The OpenSolaris Web site [5] is a great portal into Sun. The discussion forums there are lively and populated by interested users and Sun employees. Sun frequently seeks feedback there about how a specific feature should work, and about the relative priorities of specific features. Gone are the days when Sun was a black box, emitting an occasional product, with users complaining that Sun didn't listen. At least Sun is now listening (and, even more important, discussing). How this evinces itself in products is an open subject, but clearly progress is being made. Of course the OpenSolaris source code is available there as well, along with some source code tours. The ZFS tour is a thing of beauty, as is the DTrace source code itself.

Not all code is available on opensolaris.org before it ships. Recently Sun created the "Solaris 8 Migration Assistant" (S8MA, née project Etude) and launched it as a production-ready unbundled package without (yet) putting it on opensolaris.org. By the way, S8MA is a very interesting solution to the problem of having lots of Solaris 8 (on SPARC) systems and not wanting to do a full migration of the applications to Solaris 10. S8MA will do a physical-to-virtual capture of a Solaris 8 system and install it in a special Solaris 8–compatible container on Solaris 10. Several Solaris 8 environments could fit on modern Solaris 10 hardware, allowing consolidation as well as reducing rack space and power and cooling needs as those Solaris 8 systems are retired. Those Solaris 8 applications can continue running in that S8MA container until the end of life of Solaris 8 or until you make the move to run them natively on Solaris 10. As this is a new feature, my recommendation is to test it thoroughly and then plan on deploying old development and QA environments on it before attempting production conversions.

Another project that has my attention is "xvm ops center." This project ships in early 2008 and appears to be (yet another) attempt by Sun to create a Solaris administration and provisioning tool. The project is supposed to be released under the GPLv3 license, which gives it a much higher likelihood of success than previous closed-source attempts.

The Future of Sun, Revisited

Certainly, this is not your Mama's Sun. Solaris and OpenSolaris (both the distribution and the Web site) are welcome changes from the old slower-moving Sun. Signs of the revitalization of Sun are everywhere. Take, for example, the inclusion of Solaris components in other operating systems. My other favorite operating system, Mac OS X Leopard, includes DTrace, read-only ZFS, and other components from OpenSolaris. Of course Mac OS X is built on FreeBSD, which also has those features.

Another sign of the times is that Solaris is being installed and supported by Sun competitors. As of December 2007, HP, IBM, and Dell allow the purchase and preinstallation of Solaris on some of their systems. Dell has gone from one end of the spectrum to the other, offering both Solaris and the OpenSolaris distribution (once it is available, presumably!) preinstalled and supported on all of their blades and rack-mount servers. Who knows where this is all heading, but can OpenSolaris on Dell's desktops and laptops be far behind? (See the *eWEEK* article [6].)

Many IT managers don't yet seem to realize that Sun makes x86 systems (both AMD and Intel) that are certified for Red Hat, SUSE, and VMware. Unfortunately, Sun has not yet announced the inclusion of the VMware "ESX Lite" firmware in its systems, somewhat limiting them as a VMware choice. ESX Lite is a new feature as well, so Sun may include it as it ripens. The most stunning change in Sun moving from "the SPARC/Solaris Company" to "a tier-1 multi-OS systems provider" was the recent deal with Microsoft. Sun is now a Microsoft Windows OEM, allowing Sun to preinstall and provide support for Windows on all of its x86 systems.

Conclusions

It used to be easy to fill in vendor-feature checklists and cross Sun off IT vendor lists. Those days appear to be gone, with Solaris and OpenSolaris expanding their feature sets and platforms, and Sun's platforms supporting more and more operating systems. The future of Sun seems brighter than it has been since the good ol' dot-com days. (It was difficult to avoid quoting Timbuk3 here, but I managed, and you are welcome.)

Next Time

There are certainly many interesting Sun topics to cover here in the future. In the next issue security will come to the fore. There's a new security sheriff in town—and it's publishing security standards documents for many operating systems, complete with scripts to test the level of security on a system compared to the standard. Such efforts should be applauded, but how does it work in the real world? Tune in to PATS to find out.

REFERENCES

[1] http://en.wikipedia.org/wiki/Andy_Bechtolsheim.

[2] http://www.opensolaris.org/os/project/indiana/resources/getit/.

[3] http://www.opensolaris.org/os/project/indiana/resources/problem_statement/.

[4] http://opensolaris.org/os/projects/#portal.

[5] http://www.opensolaris.org.

[6] http://www.eweek.com/article2/0,1895,2216876,00.asp.

DAVID JOSEPHSEN

iVoyeur: permission to parse

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

dave-usenix@skeptech.org

HAVE YOU EVER NOTICED THAT THERE is an adversarial relationship among the services we provide, the emergent security controls we put in place to protect them, and our monitoring tools? It works like this: We install a service—a Linux box, for example—and then we want to monitor it, so we use a monitoring system with ICMP echo requests (we ping it). Then, like clockwork, along comes portknocking, a clever bit of security-related trickery to muck things up.

I once had a friend whose love life worked the same way. He'd get a good thing going, and then along would come his French ex-girlfriend to mess things all up. He knew it was coming. He could see it a mile away, but she was just so cute and clever that he couldn't ever resist (and this too he knew). He even had a name for it. He called it a "malheur à trois" (doom triangle). Eventually he moved to Arkansas (a state, I'm told, that's like kryptonite to the French).

You and I both know that we can't resist portknocking no matter what state we run to (it's *that* cool), which is why we use flexible monitoring systems. We need to be able to work around things such as security-related trickery from time to time. And if it can happen to ping, it can happen to pretty much any service we run, so I thought it would make an interesting subject for a monitoring article or twelve. But rather than bore you with ICMP, I'd rather cover something a bit more complex and practically useful.

If HTTP loses the monitoring popularity contest to ICMP, it's not by much. And being a stateless protocol, with oodles of strange and intricate authentication mechanisms, it's an ideal candidate for us to take a look at. As a bonus, HTTP follows the malheur à trois pattern perfectly. Long ago we made a bunch of simple Web sites, for which we created a bunch of simple monitoring tools, and then along came single sign-on and Web services.

The safest way to make sure a Web site is functional is to request the page and parse it for some text. This accounts for pretty much everything that could go wrong, including application server trouble and even a malfunctioning database back end. But nowadays everyone is using form-based authentication, session cookies, and magically encoded URLs to handle Web site security. It's not enough that our tools support basic auth anymore, they need to act like real users, filling out forms, making multiple requests, and maintaining application state.

In this article I'll show you how to use a personal Web proxy to dissect typical modern HTTP authentication. Then I'll get you started scripting the monitoring of your Web apps with good-ol' wget. The general idea is to capture a valid authentication session with your Web site, and then extract and replay the key elements. In short, you'll perform a man-in-the-middle attack followed by a replay attack (and without ever removing your white hat).

To play along at home, you'll need to get a Web proxy, but not a proxy in the squid sense. You'll need a special-purpose proxy that will show you the content of the HTTP requests and replies between you and the site you want to monitor. Several of these exist, and I'm not particularly fond of any of them, but the one I tend to use the most often is Burp Proxy [1], which is part of a suite of tools called the Burpsuite. Launch Burpsuite, or the tool of your choosing, and point your browser at it by configuring your browser to use a proxy. For specifics on the use of Burp Proxy, check the help file [2].

Most proxies of this type, including Burp Proxy, have something akin to an "intercept" button. When intercept is "on" the proxy will intercept requests and prompt you to either allow or deny them. For our purposes, this isn't necessary, so I advise you to turn intercept off. With intercept off, all of the requests are still captured and stored, but you aren't prompted for anything. The stored requests are available in the history tab in Burp Proxy.

The Web app security I'm reverse-engineering today is actually in use by a real publicly facing entity. I simply poked around the various services-based sites I use on a regular basis for one that had a good mix of authentication-related stuff. I've anonymized the headers in the listings to avert phone conversations with angry lawyers. For the purposes of the article, assume that we need to monitor a shrubbery management app at www.mysite.com. This site is part of a larger, landscape-related management services organization, and as such they use single sign-on at www.authsite.com, so you can manage shrubbery and a little path running down the middle without having to log in twice.

HTTP conversations, as I'm sure you're already aware, are made up of a header and data section (similar to SMTP conversations). The server and client can use the headers to talk about things such as the HTTP version number and supported features. They'll also use the headers to pass cookies back and forth. The data section is for, well, data. Obviously, where authentication is concerned most of the interesting stuff is in the header section. The notable exception is when a form is used to collect the user name and password. When this happens, we'll be interested in the POST data from the client. Generally the client will make a request of the server, to which it receives a reply. In HTTP, the server can only react to what it is asked for, so the server uses things such as HTTP redirects to influence the client when it needs to. Requests take one of two forms: GET requests and POST requests. POST requests are used for submitting sensitive information such as user names and passwords.

To keep things simple in the example that follows, I've filtered out quite a bit of extraneous stuff such as requests for graphics and style sheets. I've also summarized a bunch of requests that provided me authentication-related cookies, because they weren't necessarily relevant to our automating things later. What's left are four key transactions that we'll need our monitoring script to replay to get things working. My point in telling you this is that in real life it takes a bit of time to separate the wheat from the chaff. Be patient.

So let's get started. I'm intimately familiar with this shrubbery site, as I use it quite a lot, so I already know that to monitor the page I want, I'm going to have to fill out a form, and I already know the URL of the authentication page, but I don't start my capture there. First I load a public page to see if it passes me any cookies. Many authentication setups expect you to act like a human, and when you don't they'll redirect you somewhere that suits their needs. For example, if you show up at an authentication page without certain cookies, then the authentication code may freak out because it can't figure out what you're asking for permission to see.

Freaking out will probably entail redirecting you back to some public section of the site. Automating reactions to this kind of thing can be difficult to do. Instead, act like a human and go someplace public first the way a human would. Firing up my proxy and loading the front page, I get the headers in Listings 1a and 1b. Listing 1a shows a request for the main page of mysite, and Listing 1b shows the reply. Sure enough, the server immediately hands me a session cookie. This is a pretty strong indication that our script is going to need to save and present cookies when we monitor this site in the future.

```
GET / HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.2) Gecko/20060308 Firefox/1.5.0.2
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png
     ,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

LISTING 1A: HTTP HEADER FOR REQUEST IN A PUBLIC SECTION

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: public
pragma:
Set-Cookie: JSESSIONID=5BC21F0AC321558C088C4D13ADC35F0D;
Content-Type: text/html;charset=iso-8859-1
Date: Wed, 21 Nov 2007 17:03:22 GMT
Content-Length: 11086
```

LISTING 1B: RESPONSE TO THE REQUEST SHOWN IN LISTING 1A, WITH COOKIE

With the proxy in place, I proceed to make a request for something secure. For a few moments I'm bounced around to various pages on the site. Each of these represents some back-end application code that is attempting to determine who I am and whether I am allowed to view what I'm asking for. Along the way I pick up several more cookies and get transferred to HTTPS. One of the cookies is a monster called "s_sess," which appears to contain very specific information about what I'm asking to see. Another cookie, "s_pers," has some gobbledygook that's probably associated with who I appear to be and what level of access I currently possess. Eventually, the application decides that I can't be trusted and punts me to its parent single sign-on authority, authsite. The header of this last request, the one just before I'm redirected to authsite, is Listing 2a.

```
GET /home.jsp?cat=5 HTTP/1.1
Host: www.mysite.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.2) Gecko/20060308 Firefox/1.5.0.2
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: https://www.mysite.com/
Cookie: JSESSIONID=5BC21F0AC321558C088C4D13ADC35F0D;s_sess=%20s_cc%3Dtrue%3B%20s_sq
    %3Dauthsiteprod%253D%252526pid%25253DUS%2525253AWelcome%2525253Emysite
    %2525253shrubberyProgram%2525253APersonalShrubbery%252526pidt%25253D1%252526oid
    %25253Dwww.mysite.com/home.jsp%2525253Fcat%2525253D5_1%252526oidt%25253D1
    %252526ot%25253DA%252526oi%25253D1%3B; s_pers=%20s_dfa%3Dauthsiteprod
    %7C1195667245382%3B
```

LISTING 2A: REQUEST FOR AUTHENTICATION, WITH COOKIES COLLECTED SO FAR

```
HTTP/1.1 302 Moved Temporarily
Server: Apache-Coyote/1.1
Set-Cookie: StaticTrackingCookie=dzGTdukyUUTcrTcOGzUd; Expires=Mon, 09-Dec-2075 20:17:50 GMT
Set-Cookie: TrackingCookie=24Od2TmMzzdhvdh8O4z2; Path=/
Location: https://www.authsite.com/shrubbery/us/action?request_type=authreg_ssologin&target=https
    %3A%2F%2Fwww.mysite.com%2Fhome.jsp%3Fcat%3D5
Content-Length: 0
Date: Wed, 21 Nov 2007 17:03:42 GMT
```

LISTING 2B: RESPONSE TO THE REQUEST IN LISTING 2A

As you can see, I've presented the various cookies I received in my interaction with mysite. The reply in Listing 2b is a redirect to the authsite. Before we go, we're given a few tracking cookies for good measure. So our monitoring scripts are certainly going to need to handle cookies if they expect to play well with this shrubbery management site. We could use our proxy to withhold some of these cookies, just to see which of them are actually required by the site and which are just nice to have, but the safest thing to do would probably be to make sure our script gets all of them. This appears to be a JSP back end after all, and one never knows what those Java guys are thinking.

At authsite, we're ping-ponged around for a while, picking up more cookies in the process. Finally, we're presented with a simple form asking us for our user name and password. Listing 3a displays the POST header and data that I send to authsite. Our new cookies are presented to the form processor as well as my user name and password, which can be seen toward the end of the POST URL. The server responds with some more cookies and a 302 redirect, as seen in Listing 3b. This redirect is to another URL on the authsite, and it appears to be related to requesting SSO-related credentials to access our originally requested shrubbery-related content.

```
POST /myshrubberybbage/logon/us/action?request_type=LogLogonHandler&location=us_logon2 HTTP/1.1
Host: www.authsite.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.2) Gecko/20060308 Firefox/1.5.0.2
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
```

```
Connection: keep-alive
Referer: https://www.authsite.com/myshrubbery/logon/us/en/en_US/logon/LogLogon.jsp?DestPage=https%3A%2F
    %2Fwww.authsite.com%2Fmyshrubbery%2Fus%2Faction%3Frequest_type%3Dauthreg_ssologin%26target
    %3Dhttps%253A%252F%252Fwww.mysite.com%252Fhome.jsp%253Fcat%253D5
Cookie: s_vi=[CS]v1|474464E90000173B-A170C2800002396[CE]; SaneID=67.88.91.16-1195664628842678; s_sess=
    %20s_cc%3Dtrue%3B%20s_sq%3Dauthsiteprod%253D%252526pid%25253DUS%2525253AMYCA-Login-
    LightVersion%2525253EUserManagement%2525253Aauthsite%252526pidt%25253D1%252526oid
    %25253Dfunctiononclick%25252528event%25252529%2525257Bjavascript%2525253Aif%25252528
    %25252521checkBeforeSumbit%25252528%25252529%25252529%2525257Breturnfalse%2525253B
    %2525257Ddocument.frmLogon.submit%25252528%25252529%2525253B%2525257D%252526oidt
    %25253D2%252526ot%25253DIMAGE%3B; s_pers=%20s_dfa%3Dauthsiteprod%7C1195667265665
    %3B; s_cc=true
Content-Type: application/x-www-form-urlencoded
Content-Length: 337

DestPage=https%3A%2F%2Fwww.authsite.com%2Fmyshrubbery%2Fus%2Faction%3Frequest_type
    %3Dauthreg_ssologin%26target%3Dhttps%253A%252F%252Fwww.mysite.com%252Fhome.jsp%253Fcat
    %253D5&Face=en_US&Logon=Logon&b_hour=11&b_minute=17&b_second=32&b_dayNumber=21&b_
    month=11&b_year=2007&b_timeZone=-6&UserID=dave&Password=iheartshrubbery&x=0&y=0
```

LISTING 3A: A POST TO THE AUTHSITE

```
HTTP/1.1 302 Found
Date: Wed, 21 Nov 2007 17:04:06 GMT
Server: IBM_HTTP_Server/2.0.42.2-PK29827 Apache/2.0.47 (Unix) DAV/2
Set-Cookie: shrubberyboxvalue=d9ad1ab0-02271d96-5153a860-770139b1;Domain=.authsite.com;Path=/; Secure
Cache-Control: no-cache="set-cookie,set-cookie2"
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Set-Cookie: shrubberyboxpub=7d38d1a8936edc29f58b2859d260885e;Domain=.authsite.com;Expires=
    Fri, 13-Nov-2037 17:04:06 GMT;Path=/
location: https://www.authsite.com/myshrubbery/us/action?request_type=authreg_ssologin&target=https
    %3A%2F%2Fwww.mysite.com%2Fhome.jsp%3Fcat%3D5
Vary: Accept-Encoding
Keep-Alive: timeout=30, max=100
Connection: Keep-Alive
Content-Type: text/html
Content-Language: en
Content-Length: 0
```

LISTING 3B: RESPONSE TO THE POST IN LISTING 3A

```
HTTP/1.1 200 OK
Date: Wed, 21 Nov 2007 17:04:08 GMT
Server: IBM_HTTP_Server/2.0.42.2-PK29827 Apache/2.0.47 (Unix) DAV/2
Set-Cookie: MR=4;Domain=.authsite.com;Expires=Sat, 30-Jul-2039 18:50:49 GMT;Path=/
Cache-Control: no-cache="set-cookie,set-cookie2"
Expires: Thu, 01 Dec 1994 16:00:00 GMT
Set-Cookie: Domain=.authsite.com;Expires=Sat, 30-Jul-2039 18:50:49 GMT;Path=/
Vary: Accept-Encoding
Keep-Alive: timeout=30, max=100
Connection: Keep-Alive
Content-Type: text/html;charset=ISO8859-1
Content-Language: en
Content-Length: 363

<html>
<head>
```

```
<meta http-equiv="Refresh" content="1;
    url=https://www.mysite.com/home.jsp?cat=5&ctoken=2608C5DB4EFAAEE2B9B4BA4A0245C025062C70F042D494
    4F1AD94166EFBD3497A24EE95ADEBEE0E0&crIndex=0&crk=60387FA24B7E7BBBF7A54A08D48AC048&tier=CA&
    sid=67.88.91.16-1195664628842678">
</head>
<body>
</body>
</html>
```

LISTING 4: REPLY

When we follow the redirect, we're presented with the reply in Listing 4. This reply links us back to the shrubbery site by way of a Meta Refresh Tag. The URL in the tag is what I refer to as a "Magic URL." As you probably already know, authsite cannot give us a "yeah, he's good" cookie, since cookies can only be read by the domain that wrote them. Mysite can't read cookies authsite gave us. Instead, authsite gives us an authentication token in the URL. The magic URL should be cryptographically verifiable by mysite, should work only for us, and should be robust against replay attacks by folks pretending to be us (hence the magic). In practice it is rarely any of these things.

So how in heck do we automate all of this? In fact, it turns out to be pretty simple with the old standby, wget. This great piece of software handles cookies (if you tell it to), automatically follows redirects, and generally just does the right thing. With wget we can get from public site to SSL-enabled, protected content in three commands:

```
wget —no-check-certificate —delete-after —keep-session-cookies \
    —save-cookies mmmcookies http://www.mysite.com

wget —no-check-certificate —delete-after -keep-session-cookies \
    —save-cookies mmmcookies —load-cookies mmmcookies \
        https://www.mysite.com/home.jsp?cat=5

wget —no-check-certificate —keep-session-cookies \
    —save-cookies mmmcookies —load-cookies mmmcookies \
        -O parseme.html —post-data='request_type=authreg_ssologin&
        target=https://www.mysite.com/home.jsp?cat=5&Face=en_US&Logon=Logon&
        b_hour=12&b_minute=17&b_second=32&b_dayNumber=21&b_month=11&
        b_year=2007&b_timeZone=-6&UserID=dave&Password=iheartshrubbery
                &x=0&y=0' https://www.authsite.com/myshrubbery/us/action
```

The key cookie-related options are —keep-session-cookie, —save-cookies, and —load-cookies. They're all pretty self-explanatory. The save and load options take a filename as an argument and save cookies to, or load them from, the given file. The option —keep-session-cookies is necessary when you're dealing with JSP-style session cookies, since they won't be saved by default.

The first two commands use —delete-after to get rid of the file once it's downloaded, since we're not really interested in parsing any but the final content for errors. The last command uses —post-data to post the data we captured in Listing 3a. Once the data is posted, wget will automatically follow the redirects and meta-refresh, providing and saving cookies as necessary, finally providing a file called parseme.html. This file is the content we originally wanted, and it may be parsed to discover the state of the site.

This works great, and even lends itself to code reuse if you think ahead a little bit. The only caveat is perhaps that, because this particular POST data contains dates and times, you may have to programmatically generate them every time you run the script. This is pretty simple to do in any language you happen to fancy. More complicated authentication schemes may force you to parse tidbits out manually in interim steps, but I rarely run into something that wget doesn't just handle. If you're finding yourself doing a lot of parsing through interim HTML files for this or that, you might want to have a look at webInject [3]. It's another great tool which handles most of the error checking for you and even has a Nagios Plugin mode (but it doesn't automatically follow redirects, which is a bit of a drag).

Take it easy.

REFERENCES

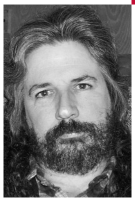
[1] http://portswigger.net/suite/.

[2] http://portswigger.net/proxy/help.html.

[3] http://www.webinject.org/.

ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is an information security geek biding his time until that genius grant finally comes through..

rgferrell@gmail.com

IN THE COURSE OF MY CURRENT employment I have cause to examine logs from a wide variety of systems. I am often struck by the utter uselessness of the so-called error codes displayed in some of these, inasmuch as no reliable and convenient means seems to exist to match the cryptic numerics with any functional description of the problem they purport to represent. Even when the harried system administrator does manage to stumble across a Rosetta Stone for decoding these mystical glyphs, the explanations are, more often than not, about as useful as a fork in a tomato-soup–eating contest. (Note to the analogy-impaired: that's not very useful.)

RodentSoft Corporation
RatsNest 2007, SP666
Misinformation Base Article #XC4-2347-0099-8675309
System Error Codes.

Code	Meaning
100	System error. Or possibly not
222	Switching to toaster-only mode
557	Bad juju
560	Could be a problem
601	Service engine soon
747	CPU not found. How are you reading this?
911	Please exit to the rear
1024	Bad code in some library or other
5555	Oopsie
6767	Sumpin' ain't right
8080	Probably time to reboot
9999	Consider getting a typewriter

UNIX isn't entirely immune from this malady, but in my experience the issue there, rather than error codes that don't tell you anything, is error messages that give you *more* information than you want to know or, at times, are willing to fall for. One of the driving forces behind this phenomenon is probably the lemming-like explosion of *nix variants available for the eager consumer to deploy and enjoy. Every flavor wants to stand out from the pack. What better way than rolling your own custom error codes, available for a limited time only with two proofs of purchase (some parts may not exist)? By my conservative estimate there are 2,357 better ways, in point of fact. But they never ask me.

I was digging around in a closet the other day looking for Christmas presents I misplaced during the

Reagan era when I came across an ancient pizza box containing my SPACK-LE-1 processor with the 0.8-MHz frontside micro-bus (and a petrified slice of pepperoni with extra mushrooms). To give you an idea of how old this computer is, the backplane was made by Sopwith.

Anyway, I decided to fire that puppy up. I broke out the priming fluid and hand crank and in no time (less than two hours) I was chugging along the information goat trail. This system, being the relic it is, had a messages log (which I had to translate on the fly using sanskrit2english.pl) that was replete with interesting errors, some of which have never before been seen in civilized society. In pursuit of esoteric knowledge and a modicum of sanity (I am, if nothing else, ever the optimist), I plumbed the depths of a file system grown fully hoar and uncovered the antediluvian artifact /usr/include/sys/errornonono.h, from which I here liberally and unabashedly quote:

```
#define EEEE      900   /* Poltergeist discovered in system       */
#define EEOW      901   /* Chassis dropped on foot                */
#define E@#*!     902   /* Monitor dropped on same foot           */
#define ENUFF     903   /* SMTP flame limit reached               */
#define EIEIO     904   /* Barnyard odor detected                 */
#define EEEW      905   /* Keyboard jammed with old tuna          */
#define EH?       906   /* Microphone input not enabled           */
#define EBERT     907   /* Streaming video rejected               */
#define ENEBRI8   908   /* Unreadable configuration data          */
#define EYEEYE    909   /* Talk Like a Pirate Day notification    */
#define EGOLD     910   /* Deprecated                             */
#define ELMER     911   /* Vendor FUD alert                       */
#define EEHAW     912   /* Extreme overclocking in progress       */
#define EMO       913   /* All display colors set to #000000      */
#define EGAD      914   /* Language file needs updating           */
#define EZZZT     915   /* Electrical short detected              */
#define ENTROP    916   /* File system randomization underway     */
#define EGON      917   /* Warning: streams crossed               */
#define ESSSS     918   /* Mouse ingested by passing snake        */
#define ENDER     919   /* Numerous bugs in system                */
#define EGGCESS   920   /* Too many shells installed              */
#define EBAY      921   /* Cheap surplus components failure       */
#define EZ2C      922   /* Extra large font selected              */
#define EARWIG    923   /* Contact exterminator ASAP              */
#define ENRON     924   /* Auditing disabled                      */
```

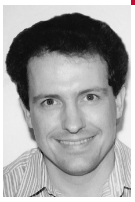
Boy howdy, that puts the "head" in "header," don't it?

N.B.: Auditing those pesky security logs does *not* consist of glancing at them briefly over a cup of double mocha decaf before you get back to Homestar Runner. You actually have to *pay attention* to logged events and look them up 'n' stuff. Maybe, if the planets are right, you might even want to *do something about* them.

This has been a public service announcement by the Council for Clearing Things Up.

NICK STOUGHTON

toward attributes



USENIX Standards Liaison

nick@usenix.org

BOTH THE C AND C++ STANDARDS ARE being revised at present, and one proposal the two revision projects have in common is to include syntax for attributes, a feature present as an extension in most modern C and C++ compilers.

Attributes allow the programmer to give additional hints to the compiler about how to generate code. They decorate variables, functions, and types. Both C and C++ have numerous places within their standards (and an enormous number, when one considers currently deployed applications) where attributes would help.

There are of course many ways to invent a syntax for a new language feature. One way is to invent new keywords in the language to represent the new feature. However, this robs from the end-user's name space and is generally regarded as a bad thing to do, unless the keyword uses an already reserved name space (which, in C, means it has to start with an underscore). Another alternative is to find some currently illegal combination of punctuation marks and make them a legal way of introducing the new feature. This cannot break existing programs . . . they wouldn't have compiled with older compilers. However, it does make it harder to use the preprocessor to mimic the new standard on an older compiler.

But, as I stated earlier, most modern compilers have *already* implemented attributes as an extension. GCC calls them attributes, whereas Microsoft's Visual C++ compiler calls them "declspec" (and almost every other compiler follows one or the other of these). In both cases, the existing practice has been, in fact, to use a new keyword. Both of them prefix their new keyword with two underscore characters, to put it into the name space reserved for the implementation.

Let's look at a trivial example of using attributes to decorate a function. I'm sure everyone who programs in C or C++ has at some time written a function something like the following:

```
void fatal(const char *msg)
{
    extern FILE *logfile;

    if (logfile) {
        fprintf(logfile, "Fatal: %s\n", msg);
        fclose(logfile);
    }
    fprintf(stderr, "Fatal: %s\n", msg);
    exit(1);
}
```

This simple function does some cleanup and exits the application on a fatal error. The function doesn't return; it calls exit(). There are a couple of things an optimizing compiler wants to be able to do with a function that doesn't return: remove dead code that follows a call to a nonreturning function and be able to notice that it doesn't need to worry about return paths following such a call. (Ever had that annoying error message "file.c:13: warning: control reaches end of non-void function"?) A function that doesn't return doesn't need to clean up the stack after itself, either.

Current existing practice in GCC allows you to add an attribute to the function prototype to indicate this:

```
__attribute__((noreturn)) void fatal(const char *);
```

The Microsoft compiler spells it slightly differently, but with the same effect:

```
__declspec(noreturn) void fatal(const char *);
```

The two committees, C and C++, are taking a very different approach to adding attributes.

The C Approach

The C committee wants to follow existing practice as much as possible; it is therefore looking at the __attribute__((xx)) and __declspec(xx) syntaxes closely. The committee will likely pick one rather than the other, and it may consider cleaning up the name a little. (All those underscores surely do look ugly!) They could go for new keywords for every attribute (e.g., noreturn) as a top-level keyword, but that would be very inflexible and hard to extend (al though there is precedent, since some of the current keywords, such as register, are really attributes). And remember what I was saying about keywords: Adding new keywords to the language is always going to be an uphill battle, as the users' name space is invaded. The syntax itself, however, is felt to be less important than the semantics of attributes. The intent of the committee is to select a common set of attributes that most vendors already support and to standardize what these attributes actually mean. To allow for further extension of this, the standardized attributes will have stdc_ prefixed to their name. The current proposal lists:

- stdc_noreturn: Applies to a function, indicating that the function does not return.
- stdc_pure: Applies to a function, indicating that the function has no side effects and will always return the same result for the same arguments (allowing the optimizer to possibly cache results).
- stdc_warn_unused_result: Applies to a function and will cause the compiler to issue a warning diagnostic if the result is not used (e.g., malloc() would be an example where this is appropriate).
- stdc_nonnull: Applies to a parameter to a function, indicating that the argument cannot be null.
- stdc_unused: Applies to a parameter to a function or to a variable, indicating that this parameter or variable is not used, but only required to ensure that the function has the correct signature.
- stdc_deprecated: Applies to a function, permitting the compiler to warn if the function is used.
- stdc_align: Applies to any variable, indicating the alignment of that variable.
- stdc_thread: Applies to any local variable, indicating that there should be a separate copy of the variable for each thread (GCC has a keyword, __thread, to do this).

- stdc_packed: Applies to a structure or union, indicating that no padding should be included, minimizing the amount of memory required to hold the type. It is also applicable to an enum type, indicating that the smallest integral type appropriate be used (e.g., a packed enum with fewer than 256 discrete values should be stored in a char).

Other attributes may yet be added to this list. In particular, the committee spent considerable time at its most recent meeting discussing the cleanup attribute from GCC, comparing it to the try {} finally {} construct added to Microsoft's compiler. A paper on this subject is expected at the next meeting, in April 2008.

The C++ Approach

The C++ committee, in contrast, *loves* to invent! If no new keywords are to be added to the language, why not invent a whole new syntax? Their proposal currently describes the syntax for adding attributes and only a few of the attributes themselves (noreturn, final, and align). The proposed syntax adds attributes surrounded by [[...]], *after* the definition. Currently both GCC and the Microsoft compiler expect attributes *before* the thing that they modify, though GCC can accept them after in some circumstances. So the fatal example above would become:

```
void fatal(const char *) [[noreturn]];
```

This syntax certainly doesn't suffer from the excess of underscores and general ugliness in the existing practice. It is certainly true that, by using the currently implemented extensions, the syntax can very rapidly get to be so opaque as to be almost unreadable:

```
int i __attribute__((unused));
static int __attribute__((weak)) const a5
    __attribute__((alias("__foo"))) __attribute__((unused));

// functions
__attribute__((weak)) __attribute__((unused)) foo()
    __attribute__((alias("__foo"))) __attribute__((unused));
__attribute__((unused)) __attribute__((weak)) int e();
```

The C++ proposal uses some aspects of the GCC syntax, but it removes that which the committee deems to be controversial. As stated, instead of __attribute__, which is long and makes a declaration unreadable, the proposal uses [[ ]] as delimiters for an attribute. For a general struct, class, union, or enum declaration, it will not allow attribute placement in a class head or between the class keyword, and the type declarator. Also, unlike the GCC attribute and Microsoft declspec, an attribute at the beginning will apply, not to the declared variable, but to the type declarator. This will have the effect of losing the GCC attribute's ability to declare an attribute at the beginning of a declaration list and have it apply to the entire declaration. The committee feels that this loss of convenience in favor of clearer understanding is desirable.

```
class C [[ attr2 ]] { } [[ attr3 ]] c [[ attr4 ]], d [[ attr5 ]];
attr2 applies to the definition of class C
attr3 applies to type C
attr4 applies to declarator-id c
attr5 applies to declarator-id d
```

Another aspect of the C++ proposal is to apply attributes to things other than simply variables, functions, and the like—for instance, to blocks and to translation units (or files). This aspect of attributes has no real implementation experience, although some compilers use the #pragma or _Pragma construct from C for something similar. So, for a global decoration or a basic statement, you might say:

```
using [[ attr1]];
```

to have attr1 apply to the translation unit from this point onward. Similarly, for a block, one might have:

```
using [[attr1]] { }
```

Now attr1 would apply to the block in braces. For a control construct, an annotation can be added at the beginning:

```
for [[ attr1 ]] (int i=0; i < num_elem; i++) {process (list_items[i]); }
```

where attr1 applies to the for control flow statement.

Conclusion

The C++ committee is also nearing the end of their revision process, whereas the C committee is just starting. If the C++ committee does indeed settle on the current proposed syntax, they will set new existing practice for the C committee to follow.

Several people have complained that recent changes to both C and C++ have led to divergence; neither committee appears to be able to follow the other's lead without making similar changes in an incompatible fashion. An example of this divergence was the introduction of variadic arguments to functions. C++ uses "..." following the last formal parameter, but in C there must also be a comma (", ..."). Indeed, some have noted that the *only* compatible extension that both languages have adopted is the // comment construct! So it will be interesting to see whether the introduction of attributes provides another place where the two languages diverge or a place where the two committees can actually work together for a change.

C is, after all, supposed to be a language compatible with C++. Once, C was a strict subset of C++, though it is no longer. But how far should they diverge? How much effort should we spend on maintaining the relationship between the languages?

I'm personally torn on the best way forward with attributes, in both languages, and would appreciate feedback.

book reviews

ÆLEEN FRISCH, BRAD KNOWLES, AND SAM STOVER

BASH COOKBOOK: SOLUTIONS AND EXAMPLES FOR BASH USERS

Carl Albing, JP Vossen, and Cameron Newham

O'Reilly, 2007. 622 pp.

ISBN 978-0-596-52678-8

REVIEWED BY ÆLEEN FRISCH

Bash Cookbook is another strong entry in the well-known Cookbook series published by O'Reilly Media. The authors have created a large collection of examples designed to address common tasks and problems as well as to educate readers about running commands and writing shell scripts under bash.

The book consists of a great many relatively short problem/task-plus-solution discussions (with related items loosely gathered into chapters). One of the strengths of the book is that these examples are placed in realistic computing contexts, so, e.g., determining the amount of time between two dates is considered with respect to NTP rather than as a coding exercise in isolation. The most extensive examples focus on text processing, simple parsing, and automating operations on files and directories. Sound best practices advice is integrated into virtually every discussion.

My initial assumption was that this book was about bash shell scripting, based on *bash* in the title and familiarity with other works in the series. However, although items involving bash scripts do comprise somewhat more than half of the book, it also contains a great deal of information that is neither bash-specific nor scripting-related. Much of the book focuses on basic UNIX commands in reasonable detail (most notably grep, find, sort, and date), along with related topics such as I/O redirection and pipes, wildcards and quoting, and search paths. The various items concerned with scripting ultimately cover a comprehensive range of relevant information, including basic script structure, file I/O, user prompting, arguments and variables, control structures and functions, script invocation methods, and security.

Several chapters in this work deserve special mention. The first chapter is an excellent tutorial for absolute beginners with bash and also includes useful information such as how to obtain bash for Windows and a list of Web sites offering free shell access.

Chapter 12 is the jewel at the center of this work. It discusses a series of very well-crafted scripts solving problems of great interest to many people: copying MP3 files to a player, creating a Web photo album for a picture directory (a great first example of generating HTML), and the like. I wish that this chapter had contained many more examples.

The book also provides a comprehensive yet compact reference appendix to all things bash: invocation options, prompt strings (including ANSI color escapes), built-in commands, shell variables, test command operators, arithmetic, and so on. I've made myself a copy that I keep closer than my bookshelf.

In general, titles in the O'Reilly Cookbook series seem to reject systematic organization and take a more exploratory, meandering pathway through their subjects. This book is no exception. I find both the ordering of the chapters and the sequencing of items within chapters very arbitrary. In addition, subjects that seem to this reader to be closely related can be separated by hundreds of pages. This is not surprising, given that the work is designed to be read in a random-access manner, like an encyclopedia. However, this design has the consequence that readers who want to explore certain topics in detail will find themselves jumping around in the text quite a bit. Fortunately, the index is excellent, so finding information is not a problem.

All in all, this is a very useful and well-written book about running UNIX commands and shell scripting in the bash shell. It should appeal to three types of readers. People who have already begun writing bash shell scripts, perhaps after reading a more discursive introductory book, will find a wealth of real-world example fragments and scripts discussed in detail. People who are looking for solutions to specific problems or techniques to specific tasks will find many helpful items within the book. Finally, people who prefer to just jump in and start trying things can use this book to learn about bash, again as both a command environment and a scripting language. Such people learn better from contextual exploration than from more abstract and linear discussions, and this book is perfect for them. All of the elements of bash scripting are included in the book, although the path connecting them is far from a straight line.

HIGH PERFORMANCE WEB SITES: ESSENTIAL KNOWLEDGE FOR FRONT-END ENGINEERS

Steve Souders

O'Reilly, 2007. 168 pp.

ISBN 10: 0-596-52930-9; ISBN 13: 978-0-596-52930-7

REVIEWED BY BRAD KNOWLES

The author of this book is the Chief Performance Yahoo! at Yahoo!. The subtitle holds a hint of the premise that instead of tuning the back-end systems for maximum performance from the perspective of the people developing on or administrating those machines, we should instead be focusing on tuning the overall system for maximum performance from the perspective of the people using the system. The author shows that Web site performance from the user perspective depends much more on the front-end architecture and on how the overall Web pages are designed and much less on the back-end throughput and how fast they can crank out the HTML. Therefore, by following the 80/20 rule, we should be working on tuning the front end and not the back end.

The author clearly explains his methodology, listing all the tools he uses in his testing, including the tools that he developed to help implement his methodology—with links for everything. In each of the chapters, the author also provides links to sample code that he has on the Web that demonstrates the technique described. Thus, the reader can see firsthand what he is talking about, and how the page is sped up by making the change being highlighted. In all, there are forty-three examples provided, demonstrating how the overall rule being discussed in a given chapter affects various different aspects of the Web page.

The book is organized according to fourteen key steps that can be taken to optimize the front-end performance of a Web site, in order of importance. However, on first glance, the particular order of the rules might not seem to make the most sense. For example, the reader might think that they should be looking at using a Content Delivery Network as one of the last things to optimize their Web site (after all local optimization has been applied). Nevertheless, by the end of Chapter Two, the reader will be convinced as to why this is the second rule. The reader might still choose to consider CDNs after local optimization, but will at least understand why CDNs are important.

Some of the chapters are very short (just two or three pages); others are longer (ten or fifteen pages). The author doesn't seem to feel the need to make any chapter longer than necessary, which results in a pretty thin book. However, although the book is packed with information, the presentation is light and easy to read. There's a full fourteen-course meal here, but each plate is as small as it can reasonably be, and each serving is already cut up into nice little bite-size chunks. This reviewer read the whole book on a flight lasting less than two hours.

The author also deconstructs the top ten Web sites on the Internet (by volume), including both his own site and major competitors, as well as some others the reader might not have otherwise expected. He is constructive when applying criticism, but he is also refreshingly honest when the competitors do well according to his methodology. Most surprisingly, he publicly applies the same type of criticism to his own site, when it does not perform as well as it could.

Some of the rule names (also used as the chapter titles) would seem to be obvious, but on further explanation the reader comes to understand the full scope of the issue at hand and how this affects the overall user experience and apparent speed of the Web site. Some of the later rules actually relate to and reexpress earlier rules, despite their inclusion in the list. Regardless of some of the apparent obvious names, most of the useful information is actually found within the chapters themselves, so there is little harm in listing the rules:

1. Make fewer HTTP requests.
2. Use a Content Delivery Network.
3. Add a (far-in-the-future) Expires Header.
4. Gzip components.
5. Put stylesheets at the top.
6. Put scripts at the bottom.
7. Avoid CSS expressions.
8. Make Javascript and CSS external (or internal).
9. Reduce DNS lookups.
10. Minimize (or obfuscate) Javascript.
11. Avoid redirects.
12. Remove duplicate scripts.
13. (Eliminate or) configure Etags.
14. Make Ajax cacheable.

The comments in parentheses are recommended alterations by this reviewer. Once the reader has completed the chapter in question, the reason for the alterations should become clear.

The author provides tips and tricks that make it obvious how some of these things can be done. For example, sites that have lots of dynamic content may think they can't implement rule #3. However, the author shows that by including the version of the object in question within the URL to the object, the front-end engineer can still add a far future "Expires:" header and make sure that the object is cached for as long as possible, while maintaining the dynamism of the site—all that is required is to switch to a different URL for a different version of that object when it gets updated.

In the case of CDNs he not only tells the reader which ones are the most commonly used and which ones are "low-cost" alternatives, he also outlines free solutions that are available. In addition, he mentions some external CDN testing services that can be used to make sure that the reader sees the global perspective on their site, and not just the very distorted picture of how it looks from the high-speed local connectivity the internal workers have from their workstations sitting right next to the servers.

The author also provides browser-specific guidance as to why the reader might want to do certain things in certain ways as opposed to other alternatives.

The one problem with this book is that it is written from the perspective of a group that has complete and total control over every aspect of their megasite, write their own tools, etc. However, most sites on the Internet today are likely to be implemented with Content Management Systems (e.g., Drupal, Joomla!, Mambo), wikis (e.g., WikiMedia, TWiki, MoinMoin), or blogging software (e.g., WordPress, Moveable Type, Bloxsom, LiveJournal), or are hosted at commercial blogging sites (e.g., TypePad, Blogger, LiveJournal, MySpace). Much of the front-end engineering for sites implemented with tools such as these will be encoded into the toolkit itself, and therefore it will be difficult to actually apply these rules.

This is not the fault of the author, but it would be very useful if a companion book were to be produced that took the Yahoo! methodology outlined and showed the reader how to implement as much of that as possible within a variety of popular tools.

Since I'm not really a Web developer or administrator myself, I don't expect to get much more out of it, so my review copy (already well-thumbed) will be handed over to the Webmaster at one of the open-source projects I help support, and I will be buying several more copies for other Webmasters and Web developers. I've definitely had my perspective on this field, and on performance tuning in general, permanently altered. I only wish someone would buy a few thousand copies of this book and freely distribute them to the key people in the various communities for Web developers, because I believe that everyone on the Internet would benefit from a universal application of these concepts.

ALTERNATE DATA STORAGE FORENSICS

Amber Schroader and Tyler Cohen

Syngress, 2007. 400 pp.

ISBN 978-1-59749-163-1

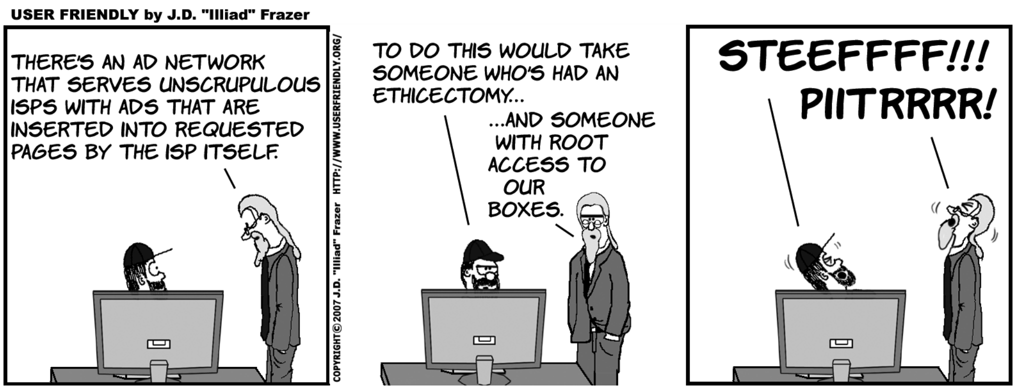
REVIEWED BY SAM F. STOVER

If you are looking for a cutting-edge book on the forensic procedure for Alternate Data Storage (ADS) devices, this is not the book for you. If you are looking for an introductory look at how ADS devices can be examined, this might be the book for you. Considering the caliber of some of the authors, I have to admit that I was a bit disappointed. I don't do forensics every day, but when I do, I prefer to have a more authoritative reference than this book provides. To be fair, much of my disappointment stems from the chapter on PDA, Blackberry, and iPod Forensic Analysis. I have some degree of experience in this area and was hoping to expand my horizons, and I was really let down. If you are interested in Blackberry hacking in particular, avoid this book altogether: 30 minutes on Google will give you far more than this chapter.

The other chapters are a bit more solid, but there is a lot of introductory text that can be found in plenty of other forensics books. I got the impression that there wasn't enough PDA/handheld-specific material to justify a $60 book, so the obligatory background filler was used to inflate the book to a final size of approximately 300 pages (not counting the index).

Now that I have all that negativity out of my system, I'd like to focus on the chapters that I did find informative. The first is Chapter 5, which addresses email forensics. As email clients become more and more advanced, extracting the actual data becomes more difficult. It's one thing to grep through someone's mutt or pine mailbox, but another thing entirely to analyze an Outlook PST file. One thing that did confuse me a bit is that the chapter starts out by outlining the exchange between a client and server, but the analysis deals solely with client systems. I was hoping for some tips on analyzing an email server, but unfortunately that was not the case. The next chapter I liked was Chapter 6, on router forensics. Again, I thought there was a little too much introductory material, but I was happy to see network infrastructure addressed in the book. The final two chapters deal with CD/DVD and MP3 forensics, respectively. The CD/DVD chapter is particularly comprehensive, and it should serve as a great reference. The MP3 chapter is also fairly complete in that it focuses on MP3 players not only as media devices but also as potential platforms for alternative uses, such as running Linux. Good stuff.

In summary, I would say that this book is a little too lean for the price tag, as only four of the chapters really piqued my interest. If you have no real experience in forensics and have a pressing need to analyze an iPod, this is probably a reasonable book to pick up. If you have some forensic background and are interested in nonstandard forensics, I'd recommend perusing the book at your local bookstore before actually ordering a copy. If you are an advanced forensic examiner, I'd wait for the next revision. I think this book has plenty of potential, but it just didn't live up to that potential this time around.


USER FRIENDLY by J.D. "Illiad" Frazer
THERE'S AN AD NETWORK THAT SERVES UNSCRUPULOUS ISPS WITH ADS THAT ARE INSERTED INTO REQUESTED PAGES BY THE ISP ITSELF.
COPYRIGHT©2007 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
TO DO THIS WOULD TAKE SOMEONE WHO'S HAD AN ETHICECTOMY...
...AND SOMEONE WITH ROOT ACCESS TO OUR BOXES.
STEEFFFF!!!
PIITRRRR!

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, networks, 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/.

DISCOUNTS on registration fees for all USENIX conferences.

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

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

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

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

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

USENIX BOARD OF DIRECTORS

2008 NOMINATING COMMITTEE REPORT

MICHAEL B. JONES AND DAN GEER
USENIX Nominating Committee

The USENIX Association is governed by its Bylaws and by its Board of Directors. Elections are held every two years, and all eight Board members are elected at the same time. Four of them serve at-large and four serve as statutory officers: President, Vice-President, Secretary, and Treasurer.

Per Article 7.1 of the Bylaws of the USENIX Association, a Nominating Committee proposes a slate of board members for the membership's consideration. As a practical matter, the purpose of the Nominating Committee is to balance continuity and capability so as to ensure that the incoming Board is composed of persons shown by their actions to be both dedicated to the Association and prepared to lead it forward.

The USENIX Nominating Committee is pleased to announce the candidates whom we have nominated for the upcoming USENIX Board of Directors election:

President: Clem Cole, *Intel*
Vice-President: Margo Seltzer, *Harvard University*
Secretary: Alva Couch, *Tufts University*
Treasurer: Brian Noble, *University of Michigan*
At Large: Matt Blaze, *University of Pennsylvania*
Gerald Carter, *Samba.org/Centeris*
Rémy Evard, *Novartis*
Niels Provos, *Google*

We are very pleased that all of these exceptional individuals have agreed to devote their time and talents to serving the USENIX Association and the advanced computing community.

Ballots and election materials will be mailed to all members in February.

SUMMARY OF USENIX BOARD OF DIRECTORS MEETINGS AND ACTIONS

ELLIE YOUNG
USENIX Executive Director

The following is a summary of the actions taken by the USENIX Board of Directors from September through December, 2007.

Member Benefits

A new policy for USENIX proceedings was approved: they will no longer be offered as a members-only benefit for the first 12 months after publication, but instead will be available immediately to everyone.

A new SAGE member benefit was approved: all system administration articles from *;login:* will be made accessible by SAGE members on the USENIX Web site.

Sponsorship

USENIX will sponsor AsiaBSDCon at a $5,000 level for 2008. Requests for sponsorship of other BSD-related conferences are encouraged.

USENIX sponsorship of the USA Computing Olympiad was increased to $20K for the 2007–2008 season, which includes hosting the USA Invitational Computing Olympiad and sending the top four competitors to the International Olympiad in Informatics in Egypt in August.

USENIX Conferences

It was agreed to co-locate the 2nd USENIX Workshop on Offensive Technologies with the USENIX Security Symposium in 2008.

Fabian Monrose was appointed to serve as program chair for the 2009 USENIX Security Symposium.

A proposal to organize a workshop on Large-scale Exploits and Emergent Threats (LEET '08) was approved. It evolved from the combination of two other workshops, the ACM Workshop on Recurring Malcode (WORM) and the USENIX Workshop on Hot Topics in Understanding Botnets (HotBots '07).

The NSDI steering committee's recommendation to have Jennifer Rexford and Emin Gün Sirer serve as program co-chairs for NSDI '09 was approved.

The Board agreed with the recommendation to have Tadayoshi Kohno and David Dill serve as co-chairs for the third USENIX/ACCURATE Electronic Voting Technology Workshop, to be held in August 2008.

A proposal from Jeff Mogul to co-locate a workshop at NSDI in April on organizing conferences for computer systems was approved.

It was agreed to continue in-cooperation status with CHIMIT in 2008.

It was agreed to accept proposals from the prospective program chairs to have the 2008 HotDep and SysML workshops co-located at OSDI. A proposal for co-
locating a workshop on supporting diversity (women and minorities) in systems research was approved. This will be a continuation/extension of a similar workshop held at SOSP in 2007.

A steering committee was formed to look into USENIX sponsoring a workshop to bring together scientists from academia and industry to encourage interdisciplinary research on multicore computing. Subsequently James Larus and Sahsha Fedorova were appointed as program co-chairs for this workshop, which has not yet been scheduled.

It was agreed that USENIX would become more active in soliciting new topics for workshops. A request for proposals was published as part of the Call for Papers for the 2008 USENIX Annual Technical Conference.

Registration Fees

In order to keep up with rising costs (especially catering) in 2008, registration fees for technical sessions at all conferences except LISA will be increased slightly (in most cases, by $15 per day) to 3-day fees of $725. LISA fees will be raised by $10 per day (to $730 for 3 days). Tutorial registration fees will increase by $10 per day for the USENIX Annual Technical Conference ($645 for 1 day) and by $20 for FAST ($245).

Miscellaneous

Peter Honeyman was appointed to be the USENIX liaison to the Computing Research Association, effective mid-2008.

Future Board Meetings

The next Board meetings will be held on February 29 in San Jose, CA (alongside the FAST conference) and on June 23–24 in Boston, MA (alongside the USENIX Annual Technical Conference.)

A subcommittee of the Board, consisting of Niels Provos, Rémy Evard, Margo Seltzer, and Ellie Young, was formed to create an agenda for a strategic planning session at the February meeting. As part of assembling data to help make this session productive, a survey of the membership was approved.

The Board thanked Hal Stern for his efforts in the past year in getting three servers donated to USENIX by Sun Microsystems.

NEW ON THE USENIX WEB SITE: THE MULTIMEDIA PAGE

ANNE DICKISON
USENIX Marketing Director

Looking for MP3s from past conferences? Want to watch the keynote from LISA '07? Check out the Multimedia page on the USENIX Web site: www.usenix.org/publications/multimedia/. You'll find MP3s of the invited talks from LISA '07, USENIX Security '07, USENIX Annual Tech '07, and LISA '06. Also available are videos from the LISA '07 and USENIX Security '07 invited talks.

conference summaries

THANKS TO OUR SUMMARIZERS

Saurabh Arora
Mukarram Bin Tariq
Leah Cardaci
Marc Chiarini
Rik Farrow
Nathaniel Husted
Kevin James
Ski Kacoroski
Kimberly McGuire
Will Nowak
Shaya Potter
Chris St. Pierre
Josh Simon
Gautam Singaraju
Anu Singh
Tung Tran

LISA '07: 21st Large Installation System Administration Conference

Dallas, TX
November 11–16, 2007

KEYNOTE ADDRESS

■ *Autonomic Administration: HAL 9000 Meets Gene Roddenberry*

John Strassner, Motorola Fellow and Vice President, Autonomic Networking and Communications, Motorola Research Labs

Summarized by Rik Farrow

John Strassner gave a keynote that was, strangely, considered to contain too much math for most of the audience. John began by demonstrating his motivation for coming up with a system that can function when there are seven different groups controlling over 60 sets of services, all—theoretically, at least—striving to satisfy the same business goals. Part of the problem with this picture (see his slide 4 diagram on the LISA '07 Web site), is that it is much too complicated for mere mortals to understand how the different groups can work together. The other issue is that the data within each group is not compatible—that is, each group is a vertical stovepipe, with systems not designed or originally intended to be shared among groups.

Even the meanings of goals, such as Service Level Agreement (SLA), are different among the various groups. At the management level, an SLA specifies the point where lowered performance means a loss of income, whereas at the network administration level, the SLA specifies the percentage of bandwidth to be allotted to each customer. The end result is that there is no single policy that works across all levels, from management all the way down to specific devices such as routers.

John's view of autonomics means that system administrators will be freed from lower-level, repetitive tasks and allowed to manage systems at a higher level. The sysadmin will not be removed from management, but from knowing how to find and tweak various configurations files. The change to the use of autonomics will be gradual, with people and autonomic systems working in partnership.

John's group, working within Motorola, has already produced working tools for managing telecommunication networks. This set of tools is designed to sense changes in a system and its environment, analyze these changes to protect business goals, and plan and execute reconfiguration. As all of this occurs, the system learns by observing the effects of reconfiguration, as well as through people providing positive reinforcement of behaviors that work. So this system encompasses machine learning as well as autonomics. And this is the point where John may have lost some of his audience, as slide 47 contained two equations (just two!), leading many people to later comment there was "too much math."

John summed up by quoting Einstein: Everything should be as simple as possible, but not simpler. Æleen Frisch then led off the Q&A by pointing out that she liked slide 40 (comparing goals from five different levels) as a concrete example. John responded that there are parallel efforts going on in his labs, and although most work gets down using CLI, all monitoring is done using SNMP—and there is no mapping between the two. He doesn't expect to see Cisco and Juniper standardize on a global lingua franca, but he said that we do need to standardize the higher-level language used to describe configuration goals (see Alva Couch's article about this elsewhere in this issue). Mark Burgess then asked how autonomics can help simplify the organizational diagram (with the seven groups) that somewhat resembles a Borg cube. John pointed out the stovepipe nature of the cube, where different groups of admins really don't talk to each other. Autonomics is about building abstractions, starting at the business details and going down to CLI.

Alva Couch pointed out that John had missed the self-protection ontology, in that sysadmins need to be able to defend themselves, that is, not be blamed for mistakes made by autonomics. John agreed, mentioning that his research system includes safety policies that prevent the autonomic system from acting before a human has reviewed the logs and potential changes. Andrew Hume asked what happens when the autonomic system has conflicting policies, as seen in the HAL 9000 killing off astronauts. John pointed out that the Policy Manager involves using many tools designed to prevent this type of conflicting policy from being created and that the learning loops are also supposed to prevent this type of thing blowing up on you. Another person wondered how new sysadmins could be taught if the autonomic system has relieved the need to perform mundane tasks. John responded that the tools they are developing will help, but that they will not solve every problem.

SECURITY VIA FIREWALLS

Summarized by Saurabh Arora (arora@kth.se)

■ ***PolicyVis: Firewall Security Policy Visualization and Inspection***

Tung Tran, University of Waterloo; Ehab Al-Shaer, University of Waterloo and DePaul University; Raouf Boutaba, University of Waterloo, Canada

Tung Tran presented PolicyVis, a tool to help manage complex policies using visualization of firewall rules and policies. He started by giving background on firewall policy management and then provided motivation for doing things a better way to help manage the complexities involved. Then he gave an overview of the PolicyVis tool, which he is developing with his professor Ehab-Al-Shaer at the University of Waterloo. PolicyVis is more than just a visual aid for policy management. It uses rectangles, colors, symbols, and notations to visualize segments and rules and supersets of investigated scope. It also supports compressing and zooming. Tung then used case studies to explain PolicyVis. These case studies included scenarios for investigating firewall policy for accepted traffic by an administrator, visualizing rule anomalies, and visualizing distributed policy configuration. He finished with an overview of the complex tasks involved in managing firewall policies, its misconfiguration, and vulnerabilities.

The PolicyVis Web site is http://www.cs.uwaterloo.ca/~t3tran/policyVis.

■ ***Inferring Higher Level Policies from Firewall Rules***

Alok Tongaonkar, Niranjan Inamdar, and R. Sekar, Stony Brook University

Alok Tongaonkar took the stage with interesting research on firewall management. He gave a problem statement of the usage of numerous low-level filtering rules which are configured using vendor-specific tools that either generate low-level firewall rules from a given security policy or find anomalies in the rules. Then he proposed a technique that aims to infer the high-level security policy from the low-level representation. The approach involves generation of flattened rules using packet classification automata (PCA).

■ ***Assisted Firewall Policy Repair Using Examples and History***

Robert Marmorstein and Phil Kearns, College of William & Mary

Robert Marmorstein began by explaining the difficulties involved in firewall repair and explained how policies are dynamic, long, and complex. Then he mentioned error detection using a passive query tool. He stressed that there is no way to automate error correction; we can only give partial specification to the tool. His technique is to use Multiway Decision Diagrams (MDD) and perform logical queries against a decision diagram model. Using the query logic, the system administrator can detect errors in the policy and gain a deeper understanding of the behavior of the firewall. The technique is extremely efficient and can process policies with thousands of rules in just a few seconds. Although queries are a significant improvement over manual inspection of the policy for error detection, they provide only limited assistance in repairing a broken policy. He gave an example of this technique on a representative packet, illustrating that the firewall complies with or (more importantly) deviates from its expected behavior.

The project is hosted on sourceforge and researchers are invited to join it: http://itval.sourceforge.net.

INVITED TALK

■ ***The Biggest Game of Clue® You Have Ever Played***

Don Scelza, Director, CDS Outdoor School, Inc.

Summarized by Nathaniel Husted (nhusted@iupui.edu)

Don started his talk by stating his objectives for the session. The session was not to teach the attendees about how to beat their kids at Clue, nor was it really about person search and search management. It was aimed more at those who were responsible for systems and were scared to death about what to do with a big one. It was about how to handle very large-scale problems. He provided examples of these large-scale problems by mentioning some of the search incidents he was involved from 2004 to 2007. Two incidents included autistic males lost in the wilderness, another included a lost woman in stormy conditions, yet another included a lost woman with a history of strokes and brain damage, and there were multiple incidents that involved females being abducted. He also mentions an IT-specific event after the World Trade Center incident in 2001 and a hacking incident in 2004.

Don then outlined the attributes of large-scale problems and their solutions. Many of the problems are time-critical. They may also involve loss of human life or of property. Some may even be criminal in nature. The solutions to these large-scale problems will generally involve numerous people. They might also involve numerous organizations and even law enforcement. Before you can solve any of these problems, you should have a plan. Even if the problem is not covered in the planning, the sheer fact that a plan was created helps you solve the problem. Don provided an example of this with a story about Captain Alfred Haines, a pilot. In 1989 his DC10 lost its hydraulic controls. Although this loss was not covered in the plans, the plans allowed him to cross off what wasn't the problem and decide how to try and land the plane safely without hydraulics.

According to Don, the three best things to know during your planning are your history, your theory, and your –subject. In the realm of lost-person search, this involves knowing what type of events have taken place in a specific area and the characteristics those events have in common. It's also good to note whether there is a common solution when similar events have taken place. Also, look at how previous problems were solved. Finally, make sure to look up and know any theories in your field that could help find the solution. You should also know your subject. In the case of lost-person search, there is a set of behaviors lost persons are most likely to exhibit.

Don then described the theory used in the lost-person search field, entailing concepts such as Probability of Area (POA), Probability of Detection (POD), and Probability of Success (POS). The POD is the probability of the searcher detecting an object if it was in a specific area. The POA is the probability that the subject is in a specific area. The POS is equal to the POA multiplied by the POD and is the probability that if the subject is in a specific area, the subject will be detected.

Don stressed that you should know your resources when solving a problem. You should know what certifications your resources have and whether they will help or hinder the search. Resources also have a cost. Finally, you need to be aware of how to get your resources if they are not currently available and how long it will take to receive those resources. In the case of lost-person search, there are ground resources, dog resources, and aircraft resources.

Don also had advice for what to do after a large-scale problem has been solved. He suggested that you review what actions were taken during the situation as well as what went well and what went poorly. The review session should also cover what needs to be changed in the preplanning stage. The review group should also decide what data needs to be cycled into history and statistics. If the situation could have been prevented, the review group should make note of that as well. However, Don warned that these review sessions can easily turn into finger-pointing sessions, so they must be implemented carefully. An example of such a review was the Hug a Tree program. This program was developed after a boy was lost in 1981 for four days. On the fourth day his body was found two miles from the campsite. The problem was that he kept moving around. A review of this situation led to the Hug a Tree program, in which young children were taught to stand still by hugging a tree.

Don ended by urging everyone to go and enact a plan when they returned to the office by posing questions such as: What history do you need to find out? What team will you put together to help you? What do you need to do when you get to the office? What preplanning and resources need to be on hand? Finally and probably the most important, Where will you get coffee?

In the question and answer period, Don was asked how an ICS would be scaled down to an organization with few individuals. Don replied that there were times when only two people were in the command structure during an incident he was involved in. One of the benefits of a good ICS plan is that the structure can be grown as time proceeds. Don said that this sort of growth was one of the benefits of preplanning.

In response to how much IT was used in search and rescue and whether there was a contingency plan, Don replied that IT is heavily used in search and rescue. People in Ops use it to print maps and people in Plans use it for spreadsheets. He also said that many things are still done by hand and when computers malfunction or something stops working, paper forms provide the needed backup.

The next questioner asked whether Don needed volunteers to help design and implement a computerized system for search and rescue. Don answered with a resounding yes and suggested that anyone who wanted to help should contact him. His email is dscelza@cdsoutdoor.com.

The next question dealt with morale and energy issues during extended searches. Don mentioned that as time progressed during a long search, he brought in counselors to sit down and talk with the individuals helping with the search. The counselors wore brown vests and acted incognito. One of the keys to maintaining morale is to keep the team briefed on the current status. The commander is the driving person who keeps everyone motivated. The key, Don said, is communication.

The final questioner asked how one might work around the fact that best practice can be a competitive advantage in the private sector. Don acknowledged that this is a problem and that corporate citizens need to figure out how to sanitize their plans so that they can provide the information to others. Don also mentioned that talking to people at conferences such as LISA is one of the best ways to share information while staying under the radar.

INVITED TALK

- ***Deploying Nagios in a Large Enterprise Environment***

  Carson Gaspar, Goldman Sachs

  Summarized by Josh Simon (jss@clock.org)

In his invited talk "Deploying Nagios in a Large Enterprise Environment," also known as "If You Strap Enough Rockets to a Brick You Can Make It Fly," Carson Gaspar discussed how a project went from skunk-works to production and how monitoring was explicitly delayed until after an incident. Their Nagios (version 1.x) installation had several initial problems:

- Performance: By default, Nagios (pre 3.x) performs active checks and can't exceed about three checks per second and did a fork()/exec() for every statistical sample. Also, the Web UI for large or complex configurations takes a long time to display (an issue fixed in 2.x).
- Configuration: Configuration files are verbose, even with templates. It's too easy to make typos in the configuration files. Keeping up with a high churn rate in monitored servers was very expensive.
- Availability: There were hardware and software failures, building power-downs, patches and upgrades, and issues of who monitors the monitoring system when it's down.
- Integration and automation: Alarms need to integrate with the existing alerting and escalation systems, and they need to be suppressed in certain situations (e.g., when a building is intentionally powered down). Provisioning needed to be automatic and integrated with the existing provisioning system.

They solved or worked around these problems by switching from active to passive checks (which gets them from 3 to 1800 possible checks per second), splitting the configuration to allow multiple instances of Nagios to run on the same server, deploying highly available Nagios servers (to reduce any single points of failure), and generating the configuration files from the canonical data sources (for example, so any new server gets automatically monitored). They also created a custom notification back end to integrate with their Netcool infrastructure and to intelligently suppress alarms (such as during known maintenance windows or during scheduled building-wide power-downs).

The monitoring system design criteria specified that it had to be lightweight, with easy to write and easy to deploy additional agents, avoid using the expensive fork()/exec() calls as much as possible, support callbacks to avoid blocking, support proxy agents to monitor other devices (such as those where the Nagios agent can't run, such as NetApps), and evaluate all thresholds locally and batch the server updates.

The clients evolved over time; some added features included multiple agent instances, agent instance-to-server mapping, auto reloading of configuration and modules on update, automatically reexecuting the Nagios agent on update, collecting statistics instead of just alarms, and performing SASL authentication among components. The servers evolved as well, with split-off instances based on administrative domain (such as production application groups versus developers), high availability, SASL authentication and authorization, and service dependencies.

The project initially involved a single project with fewer than 200 hosts but was eventually scaled up to large sections of the environment. Documentation and internal consultancy are critical for user acceptance, as is the architecture for the eventual adoption in production for the enterprise. For example, one HP DL385G1 (dual 2.6 GHz Opteron with 4 GB RAM) is running 11 instances with 27,000+ services and 6,600+ hosts, and it's using no more than 10% CPU and 500 MB RAM.

- ***Application Buffer-Cache Management for Performance: Running the World's Largest MRTG***

  David Plonka, Archit Gupta, and Dale Carder, University of Wisconsin—Madison

  Awarded Best Paper!

No summary available.

INVITED TALK

- ***Scaling Production Repairs and QA in a Live Environment (or How to Keep Up Without Breaking the World!)***

  Shane Knapp, Google Inc.

- ***Hardware Ops Release Engineering (or How I Learned to Stop Worrying and Love the Red Tape)***

  Avleen Vig, Google Inc.

  Summarized by Leah Cardaci (lcardaci@cs.iupui.edu)

Shane Knapp and Avleen Vig both related their experiences with dealing with scaling issues for Google's Hardware Operations (HWOps) group. Shane began by briefly relating his history in Google, from starting out in a tech center in 2003 to his current work in technical project management.

He then went on to describe the changes in the nature of HWOps from 1999 to the present. Until 2003, HWOps had few machines to deal with and was able to use manual processes, physical records, and noncentralized data storage. However, the group saw growth in many areas including machines, technical information to track, and employees to coordinate. The group adopted automation for key processes such as installation, centralized their data storage, and are currently developing the next series of tools.

Shane described the current workflow of machine repairs. This high-level overview followed the process from the time the machine is assigned for repair to the time it is released. He then went on to cover how HWOps was able to scale its services to deal with the enormous increase in machines and employees. One additional challenge to this process was the fact that the changes made have to be made in a live environment, so releases had to be well planned.

The initial improvements were made by looking at the key areas that had problems or were slowing down the overall process. In addition a choice was made to develop and follow the process at a high level. This level of focus allows the individual sites to follow and develop their own process at the floor level, which is important given the diversity of the various sites involved in the overall hardware repair process. There has also been a shift from the group being a black box to the rest of the company to now using in-house technologies.

Being involved in the development of HWOps has provided several insights into how to deal with the challenge of growth and deployment in a live environment. It is important to adopt standard languages and coding styles in order to allow projects to be passed on and maintained. Although it is good to lock down the key parts of the process to allow streamlining, it is also essential to build in flexibility. Planning is crucial and the process should be as visible as possible. One of the hardest lessons learned was that sometimes you have to use the solution that is available now, even if it is not the best solution. The technologies used should be chosen carefully. For example, Python is a better choice for their purposes than Perl, because it is easier to code consistently and is more readable, allowing for easier maintenance. It is important to centralize data and use a consistent scheme so that new employees can easily understand the meaning of the data. Automate as much as possible, but workflow must be understood before automation tools can be developed. Statistical analysis can help to identify areas of the process requiring additional work.

The biggest lesson learned was simply to be careful when making changes. The consequences of any change must be fully understood. Everyone affected by the changes needs to be informed that they will take place. In case something does go wrong, it is important to have a rollback plan to restore normal operation. Be thoughtful when granting rights.

Avleen Vig went on to cover his experiences working on release engineering for HWOps. Avleen has been with Google since 2005 and worked in HWOps to develop in-group tools and release engineering processes. At first, there was no release engineering in HWOps. However, with the extreme growth seen, it became necessary to adopt a formal release process.

He went on to describe the current state of release engineering at HWOps. Before a release can happen, there must be a plan for deployment, a plan for rolling it back, testing, and notes describing the changes for the end users. Each release is categorized into one of three categories: critical, important, all the rest. These categories dictate release requirements such as minimum warning time.

The timing of a release is crucial. Releasing during weekends, holidays, or other times when staffing will be light should be avoided. Notify all those affected when a release has been successfully completed as well as when something goes wrong.

A key lesson learned is that it is important not to get mired in the red tape and to allow for flexibility. For example, it is better for a crucial fix to go out on a Friday instead of the following Monday even if that goes against the practice of not deploying on the weekend.

After the talk, the group was asked whether a change control board was used for their change control review. The process had just changed to include the involvement of a formal change control board.

Avleen was asked about the burn-in hardware testing process. He replied that this involved stress testing of the hard drive, RAM, floating point unit, and other areas.

When asked about the biggest differences made in streamlining the process, the presenters replied that looking at the life of repairs for machines helped. They were able to identify machines that continually failed and replace them.

INVITED TALK

■ *A Service-Oriented Data Grid: Beyond Storage Virtualization*

Bruce Moxon, Senior Director of Strategic Technology and Grid Guru, Network Appliance, Inc.

Summarized by Will Nowak (wan@ccs.neu.edu)

The term “Storage Virtualization” is now used to describe any level of storage abstraction. Bruce Moxon helped to shepherd the audience through the fog and understand various current and future storage technologies. Bruce first took a look at conventional storage and how that works in the enterprise. Typical situations, such as overloading a single cluster node while the other nodes remain underutilized, were tackled.

By using NetApp products as a talking point, some generic solutions to common problems were illustrated. Technologies such as vFiler allow storage administrators to segregate service-specific storage into its own virtual file server instances. This abstraction enables load sharing, or easy migration in the event of an overloaded server.

Other types of virtualization, such as data virtualization, were also touched upon. Bruce gave an example of a thin client test lab at a NetApp facility in RTP. This test lab utilized blade servers and a series of NetApp filers to simulate a large client load on the filer hardware. Each blade could boot from the network, using a virtualized file system image. This allowed the total lab to use only the base file system storage cost, plus a small storage cost for client personalization. This type of virtualization provides a tremendous savings in raw storage allocation.

In the storage futures discussion, Bruce made several comparisons of Old World technology, such as the typical NFS file server, to new technologies such as the Google File System or its open source equivalent, the Hadoop File System. Bruce suggested that these distributed file systems, which take advantage of low-cost generic hardware, would continue to gain traction where they are applicable. Other interesting developments, such as storage appliance inserts, in-line encryption, and storage direction engines, were also touched upon.

The consensus of the session was to bring home the potential advantages of looking at a virtualized storage infrastructure. Abstract out your storage requirements to better serve your customers.

VIRTUALIZATION

Summarized by Shaya Potter

- ***Stork: Package Management for Distributed VM Environments***

  Justin Cappos, Scott Baker, Jeremy Plichta, Duy Nyugen, Jason Hardies, Matt Borgard, Jeffry Johnston, and John H. Hartman, University of Arizona

Scott Baker presented a new approach to package management for administering large numbers of virtual machines. Because each virtual machine is an independent entity, this provides good isolation. However, it also results in an inefficient use of resources, owing to the inability to share file system state; namely, each VM has its own disk and each disk will be cached separately by the underlying physical machine, causing increased contention for both memory and disk resources.

To solve this problem, they introduce the Stork package management system, which enables secure and efficient inter-VM sharing of file system content. Stork has two characteristics. First, its package manager, similar to tools such as apt and yum, is combined with a publish-subscribe mechanism that enables VMs managed by Stork to be automatically notified of package updates. Second, it enables packages to be stored in the “stork nest” and then shared with any VM on the same host.

When a package is installed into a system with a stork nest, it is first installed in the local machine’s file system as well as into the stork nest. Every file within the stork nest is marked with the NOCHANGE/Immutable bit, preventing it from being modified. The nest’s version is then shared with the VM by overwriting all of the package’s files, excluding files marked as configuration files, with hard links to the version of the file in the nest. As many VMs on the host can make use of the same packages, they will each use only the version that is contained within the nest, enabling efficient sharing of files in a secure manner. Stork is currently used on PlanetLab machines, and it has been shown to offer significant disk space savings for most packages. One notable exception of this is the j2re package, where a large amount of data was unpacked during the packages post-install scripts. If the files were to be repackaged in the already extracted state, this issue would be avoided. [Editor’s note: There is a much more detailed article about Stork in this issue.]

- ***Decision Support for Virtual Machine Re-Provisioning in Production Environments***

  Kyrre Begnum and Matthew Disney, Oslo University College, Norway; Æleen Frisch, Exponential Consulting; Ingard Mevåg, Oslo University College

Kyrre Begnum presented an approach to managing large numbers of virtual machines, involving notably on what physical machine they should be provisioned. This is a hard problem because system administrators need to optimize for physical machine redundancy to enable physical servers to be removed for maintenance, without compromising the ability to use virtual machines as well as remove bottlenecks resulting from resource conflicts.

To enable system administrators to solve this problem, they introduce three metrics to help determine where a virtual machine should be deployed. The first metric focuses on the amount of server redundancy. If we were to remove a physical machine from a clustered environment, could we redeploy the virtual machines contained within it to the other machines within the cluster? This is notably a problem with Xen, as it does not allow the host to over-provision the memory resource. To quantify this, they introduce the notation R/S to express the redundancy level of a cluster, where R is the number of servers currently in use within the cluster and S is the number of servers that can be removed from the cluster.

The last two metrics deal with resource conflicts. Many resources that a VM will use are shared, one important one being disk IO. If multiple VMs on a single physical machine make heavy use of that resource, their overall performance will suffer owing to contention in use of that resource. To determine where a virtual machine should be deployed, we need to know what conflicts exist between virtual machines in their use of shared resources. The Resource Conflict Matrix enables administrators to measure the level of conflict between virtual machines deployed on their servers. The final metric enables them to measure the value of conflict on a particular server with the focus on minimizing the level of conflict.

- ***OS Circular: Internet Client for Reference***

Kuniyasu Suzaki, Toshiki Yagi, Kengo Iijima, and Nguyen Anh Quynh, National Institute of Advanced Industrial Science and Technology, Japan

Kuniyasu Suzaki presented an approach for booting virtual machines over the Internet. The OS Circular framework enables a virtual machine to fetch a disk image over the Internet using HTTP and demand-page the disk blocks that are needed *as* they are needed. These blocks will then be cached locally so that they do not have to be constantly refetched.

To enable this demand-paging model, OS Circular divides a file system image into 256-KB compressed blocks, where each block becomes its own file, named by the SHA1 hash of its data. This enables the VM to verify that the data was fetched correctly. Each file system has a mapping file that maps block numbers to the correct SHA1 named file; a file system is mounted by making use of the mapping file and demand-paging and caching the blocks as needed. A file system can be updated by creating new SHA1 named files for the updated blocks and updating the mapping appropriately.

One problem with demand-paging a file system is that network latency can have a severe impact on the file system, especially on an initial boot of it, when no data is cached locally. To optimize latency, they leverage ext2optimizer to profile the file system and place files needed by the boot processes to be placed at the beginning of the file system. By removing fragmentation normally existing in a file system and leveraging read-ahead techniques, one can minimize the overhead from the network latency.

- ***Secure Isolation of Untrusted Legacy Applications***

Shaya Potter, Jason Nieh, and Matt Selsky, Columbia University

Shaya Potter presented an approach to contain independent services and their individual application components. Software services need to be contained because software is buggy and those bugs can result in security holes, providing an attacker with access to the system. However, services are made up of many interdependent entities, so containing those entities appropriately can be difficult.

To resolve these issues, Potter et al. introduce two abstractions, Pods and Peas. Pods provide a lightweight virtual environment that mirrors the underlying operating system environment. Processes within a Pod are isolated from the underlying system, and as such Pods are able to isolate an entire service. Because a Pod is hosted on a regular machine, it does not need many of the resources regular machines need (e.g., what's needed for booting), enabling it to contain just the resources needed for the entire service.

The second abstraction, the Pea, enables a simple access control mechanism on the resources made available to the Pod. The overriding principle is that just because a process is within the Pod does not mean it needs access to the resources the Pod makes available. Peas are notable, when compared to existing containment systems such as Janus and Systrace, for performing file system security in the correct location, namely the file system itself, and therefore they do not suffer from common "time of check, time of use" race conditions. Peas also implement a simple-to-understand configuration language that leverages the skills system administrators and users already have to perform as part of their daily tasks. Finally, access control rule creation can be difficult because the knowledge necessary to build rules is divided between the developers, who know the minimum needs of the application, and the administrator, who defines local security policy, so Shaya Potter demonstrated a rule composition mechanism that enables a developer to provide a minimal rule set that defines the minimal needs of the applications while enabling the administrator to build upon that and to define what local policy one wants to apply to the application.

INVITED TALK

- ***Who's the Boss? Autonomics and New-Fangled Security Gizmos with Minds of Their Own***

Glenn Fink, Pacific Northwest National Laboratory

Summarized by Marc Chiarini (marc.chiarini@tufts.edu)

In this talk, Glenn Fink tells us that autonomic computing (AC) is coming, albeit much more slowly than we think. He also suggests that our jobs are not in danger in the near future, though a sysadmin's duties will change significantly as the world transitions to AC technologies. To put things in perspective, Fink gives us a "personal guesstimate" of how far along we are on the four big cornerstones of autonomic computing as defined by IBM: self-configuration at 60 percent (with tools such as Cfengine, Puppet, and BCFG2 aiding this process); self-healing at 25 percent (an admittedly generous estimate, because most of the academic work on this has been in the security arena); self-optimization at 10 percent (another generous estimate, as we only know how to do this in very specific domains); and finally self-protection at 40 percent (where there has been a lot of good research into detecting and responding to attacks and general failures). Of course, these progress markers do not average out to 33 percent for the whole of

AC, because we have no clear way as yet of integrating the various systems that implement these processes.

Fink presents autonomic computing as a direction (or continuum) rather than a goal. This is to say that it will always be difficult to draw a bright line between AC and non-AC systems; we will be able to watch the changeover happening, but we won't be able to "see it happen." Like many other evolutionary processes, autonomic computing is being driven by the necessity to meet demand for services. IT infrastructure growth has been exponential in recent years. Combined with a software crisis (over budget, beyond schedule, buggy, and difficult to maintain), a hardware crisis (in which volume overtakes reliability), a tech education crisis (a lack of qualified high-tech workers), and the (relatively) prohibitive costs of IT personnel, this growth rate is unsustainable without automation or excessive outsourcing. Unless we want nearly everyone in IT to lose their jobs, we need to think hard about how to build autonomic systems.

When we start deciding what AC should look like, we quickly fall into a contest between the purist and the pragmatist. The purist believes that maintenance is the dominant long-term cost, that system populations should be as homogeneous as possible, that policy should be centrally defined, and that admins should be significantly constrained to avoid conflict with autonomic processes. By contrast, the pragmatist thinks that downtime is the dominant cost and decentralized quick fixes (by almost any means) in a highly heterogeneous environment are the way to go. Fink suggests something in the middle: Ensure that pragmatic fixes feed back into an established, inspected, and trusted library of practices that is open-sourced. All autonomic computing will be done under human supervision, with the added goals of communicating why decisions were made and how those decisions relate to other autonomic systems.

Fink spent the last half of the talk enumerating both a wish list and a fear list concerning autonomic computing. In his conversations with colleagues and IT professionals, he discovered three characteristics that will be most important: AC systems should act like a junior sysadmin, investigating and reporting with lots of little open-ended tasks; they should be able to robustly handle real-world situations with little or no supervision; and they should be able to communicate like a human, providing sufficient detail in a natural language context. Of the prodigious list of fears, the most important were probably issues of trust, process verification, and delegation. How do I know the system is doing what it should? Can I trust the system to verify existing agreements or negotiate new agreements with other systems?

In the end, Fink believes that our jobs are in danger, not from autonomics, but from outsourcing. Autonomics will be able to take care only of well-defined tasks and problems, and someone will always be needed to verify autonomic behavior and adherence to business objectives. The ways in which AC will change the profession are manifold: Computers will be trusted with more kinds of work, resulting in fewer tedious tasks; sysadmins will have more time to help users (hone those social skills now!); there will be natural dividing lines among AC specialists (as witnessed in the medical fields), and ultimately it is the specialists (e.g., DB and storage) who will be impacted more than the nuts-and-bolts system and network administrators. Finally, much more ethnographic study of both IT professionals and users will be necessary before AC is ready for prime time.

INVITED TALK

■ *No Terabyte Left Behind*

Andrew Hume, AT&T Labs—Research

Summarized by Josh Simon (jss@clock.org)

Andrew Hume discussed the disk dilemma: Space is cheap, so users want, get, and use more of it. However, this leads to all sorts of interesting problems, such as how to partition and how to back up the disk (especially when you get toward terabytes on the desktop). Traditional tools (such as dump) take 2.5 days to back up 250 GB. Making the space available from servers can be problematic (given local or networked file systems and the associated problems with network bandwidth). We've talked about these issues before, but there are still no good solutions.

Let's take a hypothetical example of recording a TiVO-like service without any programming wrappers. Recording everything all the time for both standard and high-definition programming leads to about 1.7 petabytes per year of data, even assuming no new channels get added. This is too big for the desktop, so we'll need to use space in the machine room: a 2U or 3U generic RAID unit at 2–4 TB/U costs up to $1,500/TB, and you'd need 133 of them per year. This uses 16 TB per square foot and requires 27 feet of aisle space per year with modest power and cooling. But that's a lot of money and space. We can possibly be clever by looking at the access patterns; for example, we can move the older and less-accessed shows off to tape, or keep only the first 5 minutes of the show on disk and the rest on tape, and thanks to a tape library (e.g., an LTO-4 with 800 GB/tape and 120 MB/s sustained write at 60-s access and a 2.5-PB library costs $172/TB and uses 41 TB per square foot, and expansion units are $7/TB and 79 TB/square foot) we can still provide every TV show on demand with no user-visible delays. Sounds good, right?

Wrong. It gets worse when you realize the fallibility of media. Ignoring the issues with tape (such as oxide decay, hardware becoming obsolete, and so on), we've got problems with disks.

Here's the reality about using disks, networks, and tapes: Things go bad, trust nothing, and assume everything is out to get you. You don't always get back what you put out. Compute a checksum for the file every time you touch it, even when it's read-only. Yes, it's paranoid, but it's necessary if you really care about data integrity, especially with regard to disk and tape. Andrew is seeing a failure rate of about one uncorrectable and undetected error every 10 terabyte-years, even in untouched, static files.

As disk use grows, everyone will see this problem increasing over time. The issue of uncorrectable and undetected errors is real and needs attention. We need a way to address this problem.

PLENARY SESSION

■ *The LHC Computing Challenge*

Tony Cass, CERN

Summarized by Leah Cardaci (lcardaci@cs.iupui.edu)

Tony Cass discussed the challenges associated with working toward the debut of CERN's Large Hadron Collider (LHC), which will deploy next year. Cass began with an introduction to CERN and LHC. CERN's goal is to "push back the frontiers of knowledge" by investigating important scientific questions. (For example, one major question is why certain elements are heavier than others; one theory is the existence of the Hick's field.) This often involves the deployment of new technologies to support the research performed. CERN's goals are to unite people from different countries and cultures and help train future scientists.

Cass gave a brief overview of four LHC experiments: ATLAS, CMS, ALICE, and LHCb. Each of these experiments will produce about 40 million events per second, which will be analyzed and reduced to a few hundred good events per second. This means the four experiments will require around 15 petabytes of storage per year. The three steps of data handling are reconstruction, analysis, and simulation. CERN is responsible for reconstruction and data retention; other locations deal with analysis and simulation. Overall, enormous computing resources are required. The challenges involved in running these experiments are having sufficient computing capacity, managing the high number of machines required, tracking and distributing the data, and understanding the state of the resulting highly complex system.

A three-tiered system is used for data handling. Tier 0 is the accelerator center, responsible for recording and processing the data from the accelerator and long-term storage. Tier 1 centers are responsible for distributing the data to researchers, as well as for analysis of the data. Tier 2 centers are involved in simulation and end-user analysis. Grid technology was adopted to provide the high amounts of computing resources needed. This involves three grid infrastructures, EGEE, OSG, and NorduGrid. The project had to meet certain levels of interoperability for submission of jobs through the system and administration of the system. Reliability will be a continuing challenge once the experiment is launched and the project has increasing reliability goals to meet.

Management of machines is provided by ELFms Vision, a custom toolkit developed by CERN and others to provide a system that would meet all of the project's needs. Quattor provides scalable installation and configuration management. Lemon provides monitoring, including looking at information outside of the individual computers, such as UPS status. LEAF, a collection of high-level workflows, automates tracking nodes' physical status as well as their configuration status. Integration with Quattor and Lemon allow for a great deal of automation in the management of nodes. This design has allowed CERN to deal with the great increase in machines added throughout the preparation, and it will continue to scale further. A huge amount of data has to be stored and distributed for this experiment, which poses another challenge. The accelerator produces an average of 700 MB per second of data, and the system will need to be able to support almost twice that amount to allow for recovery. There are three different types of access use cases: sustained transfer to a remote site, rapid transfer of data set to nodes, and long-running analysis access of data on a server. Each type has its own requirements and creates a different footprint on the data servers. No existing system met all needs, so CERN developed CASTOR, the CERN Advanced STOrage system. CASTOR is based on databases, schedules the data distribution to prevent overwhelming the system, and also schedules based on priority. Continuing challenges will be keeping the data lifetime long enough, dealing with the disparity of capacity vs. IO rates, integrating different data systems without interfering with the use of the system, and handling the export of data.

The final challenge is to manage the incredibly complex system developed to support the LHC experiments. This has been aided by the use of a user status view, which shows the current status of all the jobs for a single site, pulling the information from the (possibly many) nodes they are running on. This also involves grid monitoring and a new visualization technique to help managers focus on the critical problems in the system.

Overall, the project involves many challenges related to its size and complexity. So far, many of these challenges have been met, but the real test will begin once the system goes into full operation.

Cass was asked whether the group was ever in a situation where waiting to buy machines would be more cost-effective. He replied that they had seen those situations, but at their scale there was a greater latency because of deployment time, so that had to be take into consideration.

Another question was whether CERN was concerned about malicious attempts to corrupt the data. Cass replied that they didn't think the project was high-profile enough for their data to be a target, but they had considered that their computing resources could be a target.

MISCELLANEOUS TOPICS I

Summarized by Marc Chiarini (marc.chiarini@tufts.edu)

■ ***Policy Driven Management of Data Sets***

Jim Holl, Kostadis Roussos, and Jim Voll, Network Appliance, Inc.

IT departments frequently ask Network Appliance for a unified software and hardware infrastructure that will provide them with easily managed and well-protected storage and data services. The primary reason for this request is to optimize the use of physical resources and reduce complexity, thereby reducing cost. A typical way to achieve this goal might be to use shared storage arrays that allow multiple disparate disks to be viewed and acted upon as single logical entities. Unfortunately, organizations rarely use a single vendor for their storage infrastructure, and even when they do, there exist incompatibilities among products and service tiers. There are frequently too many individual storage containers because of data growth and replication, making management very difficult and resulting in under- or over-provisioning of both storage and protection. Instead of having a unified physical storage and data management layer, customers tend to engage in two separate disciplines: storage management (e.g., how many and what kinds of disks, controllers, and LUs are needed) and data management (e.g., how resources are used, backup discipline, replication discipline, where to place files, databases). Since data management relies on storage management, large organizations often end up manually translating the former into the latter by way of the help desk. Roussos's team developed software to handle the automatic right-sizing and placement of storage resources.

The unified storage and data management software presented in the paper introduces three abstractions: a resource pool, a data set, and a policy. A resource pool is a fixed amount of physical capacity, performance, and IOPs along with well-defined sets of capabilities, such as deduplication, replication, and redundancy. It allows easier management and optimization across more storage containers. A data set is a collection of data and the replicas that use a single data management policy. Data sets abstract storage containers and locations from the data and reduce the number of objects to manage. A policy describes how a data set should be configured with regard to protection and provisioning. Policies establish clearly defined roles, with storage architects constructing them, data admins selecting which ones are used, and a conformance engine configuring storage according to the selected policy. The conformance engine performs multiple tasks, including monitoring current configurations, alerting administrators to policy violations, and reconfiguring automatically when possible.

Roussos gave a very compelling comparison between a traditional graph of storage infrastructure and a view of the same graph in terms of data sets, which greatly simplifies and clarifies things. The take-away from the presentation was that a unified data and storage management layer vastly reduces the number of entities that must be managed and the number of steps required to perform traditional tasks. Lastly, it gives admins the advantage of conformance monitoring, to continually check that everything is laid out according to plan.

■ ***ATLANTIDES: An Architecture for Alert Verification in Network Intrusion Detection Systems***

Damiano Bolzoni, University of Twente, The Netherlands; Bruno Crispo, Vrije Universiteit, The Netherlands, and University of Trento, Italy; Sandro Etalle, University of Twente, The Netherlands

For those system administrators who are not quite familiar with the security aspect of our profession, IDSes (or Intrusion Detection Systems) are software systems (sometimes coupled with hardware) that are designed to detect (and sometimes take action against) malicious activities occurring on a host or in a network. ATLANTIDES deals exclusively with attacks in a network. There are two approaches to detection: signature-based approaches, which search network packets for specific predefined and well-known sequences of bytes, and anomaly-based approaches, which gather statistics about the packets "normally" seen on the network and indicate when those statistics stray significantly from the norm. Network IDSes are considered an efficient second-line defense (after firewalls) because they are virtually transparent to the monitored network and generally do a decent job. There are some significant disadvantages to both types of detection, however, that can greatly reduce the cost/benefit ratio: Signatures must be carefully selected for a particular site in order to reduce the number of false alarms that are generated; anomaly-based detection uses a threshold to raise alarms, which must also be tuned. In short, these tasks threaten to overwhelm IT security personnel.

Bolzoni's team proposes a solution that greatly reduces the management workload resulting from required detection tuning and verification of alerts. ATLANTIDES is an anomaly-based network IDS that can be combined with any traditional NIDS to efficiently improve the rate of false positive alarms. Instead of watching incoming network traffic for signatures or anomalies, the system learns over a short time (one to seven days depending on the diversity of outgoing traffic) what "normal" output traffic looks like. Whenever the incoming NIDS would normally raise an alert on suspicious activity, the ATLANTIDES correlation engine determines whether the output traffic seems suspicious as well. If so, an alert is raised that, because of this double-checking, has a high likelihood of being true. If there is a mismatch between what the input NIDS sees and what ATLANTIDES sees, the system can be configured to either discard the alarm as a false positive or, in the case of a potential false negative (incoming traffic looks OK, but outgoing does not), escalate the severity of the alarm.

To determine the efficiency and accuracy of ATLANTIDES, Bolzoni's team ran tests against both a well-known Internet traffic data set (DARPA99 multiprotocol) and a recently captured unfiltered HTTP traffic data set. In the case of the DARPA data, the tests showed a reduction of between 50% and 100% in the false positive alarm rate when compared to use of a single NIDS alone. In the HTTP traffic set, ATLANTIDES also improved the rate by more than 50%. The observed maximum output analysis rate was around 100 MB/s. The team plans to do further testing with more real-world data in the near future, but they are very excited about the results so far.

- ***PDA: A Tool for Automated Problem Determination***

  Hai Huang, Raymond Jennings III, Yaoping Ruan, Ramendra Sahoo, Sambit Sahu, and Anees Shaikh, IBM T.J. Watson Research Center

Ruan presented a system that improves the efficiency with which system administrators can analyze and respond to trouble tickets. The motivation for this research was a lack of robust, tailored, and easy to use tools for problem determination. System administrators (yes, even folks at IBM) tend to troubleshoot in an ad-hoc, time-consuming manner; they build different customized scripts for different platforms and frequently reinvent the wheel; the knowledge they gain is usually stuck in their heads and cannot be easily leveraged.

The PDA approach is threefold: attempt to characterize the nature of real-world problems by analyzing problem tickets and their resolutions; provide a common platform to standardize monitoring and diagnosis tools (also known as probes); and capture problem determination knowledge in expressible rules. The approach collects both high-level system vitals and "drill-down" problem analysis steps. The study utilized about 3.5 million trouble tickets generated over 9 months and analyzed the ticket distribution and time spent resolving tickets across a wide range of products and within the products themselves.

Ruan's team discovered several interesting statistics: 90% of the tickets resulted from trouble within 50 applications, the top two being a mail app (20%) and a VPN app (10%). Within the mail app, 70% of the tickets came from only 11% of its modules. Within the VPN app, 70% of the tickets came from only 8% of its modules. More important than this distribution of trouble tickets across applications was the amount of time it took to resolve OS problem tickets, roughly an order of magnitude longer on average. Taken in combination, application and configuration problems related to problems with a particular OS made up the majority of tickets. Thus, PDA is designed to focus on issues stemming from OS and system software misconfiguration.

PDA implements a thin probe model, in which generic checks are made on managed servers on a scheduled basis. The probes can be built via native commands, scripts, existing tools, or even specialized executables. The probes transmit standardized key/value pairs to a rules engine that checks potentially extensive yes/no decision trees for compliance, asking for more probe information when necessary. If a violation is discovered, the engine executes whatever action was specified in the rule sets, which might entail terminating future probes, sending an alert to a Web interface or email, or taking corrective action. New probes and rules can be authored via a simple Web interface that leverages existing collections of trouble tickets, probes, and rules.

In future work, Ruan's team hopes to address issues with the security of authored probes and also investigate the possibility of making probes and rule sets shareable across different platforms and sites.

INVITED TALK

- ***Experiences with Scalable Network Operations at Akamai***

  Erik Nygren, Chief Systems Architect, Akamai Technologies

  Summarized by Shaya Potter (spotter@cs.columbia.edu)

Akamai deploys a large, worldwide-distributed network that provides many services, including HTTP/HTTPS, live and on-demand streaming, and app delivery. Akamai is such an integral part of the Internet that we use it every day without even realizing it.

Akamai distributes its servers all over the world, as those who use the Internet are highly distributed as well. According to Akamai's measurements, one has to be on 1000 separate networks to be close to 90% of Internet users. By distributing servers toward the edges, they gain greater performance and reliability and are able to absorb traffic peaks better, as they avoid congestion points that occur where networks peer. In fact, ISPs want Akamai, as it saves them money because the traffic never leaves their network.

To distribute content to its distributed machines, Akamai deploys its own overlay network to create a highly reliable tunnel among the machines. Today the tunnel includes 28,000 machines in 1,400 locations. As Akamai uses commodity machines and network links, it expects lots of faults, so it has to treat failures as a normal occurrence. The primary way of dealing with this is with large amounts of redundancy. Redundant machines can be easily repurposed, enabling Akamai to handle faults even within a single cluster of machines. For instance, in a single cluster of machines, a "buddy" of a machine that goes down can take over for it by simply grabbing the IP of the failed machine and handling requests that are directed to it. Geographic and network redundancy combined with multipath communication in its overlay network enable Akamai to handle faults within the network links. Finally, the company has fully redundant NOCs distributed around the world, so that no one NOC has functionality that cannot be replaced by another NOC.

To manage all these computing systems, Akamai has implemented a query system that enables efficient real-time monitoring of its systems. It uses a relational database model, in which each machine provides a set of tables that contains information about its current state. Akamai's query systems compose the information provided by the machines into a set of 1400 distinct tables, with table updating occurring in the 1–3-minute range. This enables alerts to be created via regular SQL queries and the management of a large number of machines in a more automatic manner.

INVITED TALK

■ *Ganeti: An Open Source Multi-Node HA Cluster Based on Xen*

Guido Trotter, Google

Summarized by Will Nowak (wan@ccs.neu.edu)

Guido Trotter gave an overview of Ganeti, outlining its goals and usage, provided a road map for the future, and made a valiant attempt at a live demo. Ganeti is a open source management layer that rides on top of a vanilla Xen setup, allowing management of multiple nodes in a cluster. Tasks such as provisioning, management, failover, and some disaster recovery are handled by the Ganeti software package. Ganeti's goals are formulated much like other Google technologies. The project aims to increase availability, reduce hardware cost, increase machine flexibility, and add a layer of service transparency. Ganeti was also designed to scale linearly, be hardware agnostic, be broadly targeted, and maintain small, iterative development.

Ganeti leverages Xen currently, but Guido mentioned that in the future they hope to support other virtualization technologies. The toolkit is written in Python, using LVM, DRBD, and MD for storage and Twisted with SSH for RPC. Ganeti is best supported on Debian-based systems, but porting to other Linux distributions should be trivial.

Questions were raised regarding overlap with the Linux HA project. Guido's response was that Ganeti was designed at Google internally to fit a specific need that available software could not fill and that he would be interested in seeing how the two products could better serve each other.

More information on Ganeti can be found at http://code.google.com/p/ganeti/.

MANAGING GRIDS AND CLUSTERS

Summarized by Saurabh Arora (arora@kth.se)

■ *Usher: An Extensible Framework for Managing Clusters of Virtual Machines*

Marvin McNett, Diwaker Gupta, Amin Vahdat, and Geoffrey M. Voelker, University of California, San Diego

Marvin explained the motivation of their research, which was to help system administration become more effective and allow sharing among multiple resources efficiently. Their approach is to use virtual clusters (i.e., to deploy multiple VMs on each physical machine). The tool they are developing, called Usher, simplifies VM administration. The best part about Usher is its extensible architecture. Marvin and his team have done extensive work in making Usher extensible, by providing user application APIs and VMM wrappers and using plug-ins to add new functionality to Usher. The available plug-ins as of now are LDAP, IP Manager, and DNS. Usher has been successfully deployed in the following places: the Russian Research Center at the Kurchatov Institute, UCSD CSE System, and research projects such as spamscatter and spaceshare. The Usher Web site is http://usher.ucsd.edu.

When asked whether Usher is available for all virtualization technologies, Marvin replied that it is only available for Xen, but you can easily write a wrapper for vmware, KVM, etc.

■ *Remote Control: Distributed Application Configuration, Management, and Visualization with Plush*

Jeannie Albrecht, Williams College; Ryan Braud, Darren Dao, Nikolay Topilski, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat, University of California, San Diego

Jeannie Albrecht gave an overview of building distributed applications and introduced us to the Develop-Deploy-Debug cycle of a distributed application. Then she focused on challenges involved in this cycle of locating and configuring distributed resources. She also stressed the challenges involved in recovering from failures in a distributed deployment. The goal of her research is to develop abstractions for addressing the challenges of managing distributed applications. She took the specific example of developing a distributed application, say Bytetorrent, for the presentation. She started with different phases of the application and discussed evaluation through management architecture such as PlanetLab. She explained the hurdles involved in each phase of the example application, and here she proposed a distributed application management infrastructure—Plush. She explained the Plush architecture and how it can acquire resources, configure resources, and start and monitor applications. Plush has a beautiful graphical user interface, called Nebula, that is used to describe, run, monitor, and visualize deployed applications. The Plush home page is http://plush.cs.williams.edu.

■ *Everlab: A Production Platform for Research in Network Experimentation and Computation*

Elliot Jaffe, Danny Bickson, and Scott Kirkpatrick, Hebrew University of Jerusalem, Israel

Everlab was spawned from the EU-funded research project Evergrow, which was proposed for large-scale network management. Elliot Jaffe began by giving an overview of Evergrow. During that project, they felt the need for a better management system, so they moved toward PlanetLab, which is very tightly secured and offers centralized management. But the consortium of the EU project was not very supportive in joining PlanetLab, so they came up with Everlab. He mentioned that Everlab is inviting researchers to join and use its underloaded resources (as opposed to the overloaded resources of PlanetLab). Elliot came up with a few noteworthy conclusions about research projects in general: (1) funding is only for research; (2) release, deployment, and management are not research; (3) there is a difference between a flash-in-the-pan system and a computing standard. He then asserted that sound funding should be made available for deployment and management as well.

The Everlab home page is http://www.everlab.org.

INVITED TALK

▪ *Using Throttling and Traffic Shaping to Combat Botnet Spam*

Ken Simpson, Founder and CEO, MailChannels

Summarized by Leah Cardaci (lcardaci@cs.iupui.edu)

Ken Simpson gave an overview of his approach to fighting spam, which is based on the concept of attacking spam by attacking the economics of spam. He began by relating his personal work history on dealing with spam, from his beginnings with ActiveState to forming a company with other former ActiveState employees.

Simpson went on to provide a history of the spam problem, noting that his was a rough view and anyone was free to correct mistakes during the Q&A session. In 2002, spam had not been a major problem, was not a crime in most areas, and was handled using regular expression filters. In 2003, spam had made mail almost unusable, the CAN-SPAM act was created, and the spammers went underground in response. In 2004 Bill Gates announced that spam would be beaten in two years. Now spam is a fully criminal endeavor, run by organizations such as Russian gangs.

Covered next was the economics of spam. Simpson suggested that the current way of handling spam, filters, will not be able to have an impact on the overall economics of spam. Although current filters are fairly accurate, the ease in increasing the volume of messages means that the spammers can always win the game of averages.

Currently, spam is being sent from compromised computers, which are organized into botnets controlled by a bot herder. The botnets are rented to the spammers by the bot herder, providing a constantly changing set of machines from which to send messages and thus overcome blacklisting. This doesn't mean that blacklisting is not useful; in fact it allows a great deal of spam to be blocked and keeps systems from being overwhelmed with traffic. Also, the use of blacklists now means that a given botnet will quickly lose its ability to spam, and new machines must be compromised constantly to keep up.

Botnet herders are only paid once the final SMTP acceptance message is received, so they will not profit if the mail is blocked by a blacklist or filtered. For this reason, spam software has an extremely short timeout compared to legitimate mail servers, which follow the three minutes recommended in the RFC.

The current state of affairs is that spam filtering is reaching the limit of its possible increase in accuracy, and identifying zombies to simply block traffic from them is very difficult. Simpson suggests a new approach designed to attack spam by removing the profit in it. This approach uses both blacklisting and whitelisting, and then throttles all suspicious traffic to see whether it will reach the very short timeout of the spam software.

Simpson went on to discuss a case study of the deployment of this system. In this case, a pharmaceutical company saw a overnight reduction from 70% of mail being spam to 20% being spam. The system is deployed in software at the edge of the network. One challenge introduced by this system is the fact that the throttling of suspect traffic requires a great increase in the number of concurrent connections the mail servers must handle. The solution was to introduce a system in front of the mail server that handles the throttling and to use real-time SMTP multiplexing to reduce the connections the server has to handle. Looking at the suspicious traffic revealed that 80% of those machines that dropped the connection were running a consumer version of Microsoft Windows.

Simpson was asked whether the spam software won't simply be adjusted to increase the timeout window once this approach was widely accepted. He replied that there is typically a long time before spammers adjust to such measures, and that this would still affect overall profitability owing to the short time before the machine is placed on a blacklist.

Someone pointed out that people did in fact care about spam in 1995. He went on to point out that spammers can react very quickly to changes in spam defense.

Another audience member suggested that the current profit margin was so high that it seems unlikely that serious damage can be done to the profitability of spam.

MISCELLANEOUS TOPICS, II

Summarized by Marc Chiarini (marc.chiarini@tufts.edu)

▪ *Master Education Programmes in Network and System Administration*

Mark Burgess, Oslo University College; Karst Koymans, Universiteit van Amsterdam

In this talk, Burgess discussed the philosophical and technical difficulties of supporting a traditionally vocational subject within a strong academic framework. One of the biggest controversies involves the question of teaching what is viewed not as a discipline, but rather as a set of technical skills. How do we teach something that most people believe is only gained through experience? Who should teach it, professors or practitioners? What material should be used? One quickly becomes mired in a plethora of questions for which the promise of a good answer does not even exist. Burgess suggests the following: that the "discipline" needs to be described in a fairly rigorous form that can be handed down for posterity. It cannot be presented as currently practiced, because it changes too fast; we need to find paths to and from other disciplines that will promote an awareness of the subject; we should try to preserve the hands-on, engineering-focused aspect of system administration; it is important to stay in touch with rapid industrial development (something for which other academic disciplines are not well known); and finally and perhaps most importantly, we need to make it well known that the formalization of system and network administration does not belittle those who have learned the subject by other means. We should not think of system administration in universities as replacing everything that people have learned the hard way, but rather as a way to document those efforts and hand down the best parts.

In the second half of the presentation, Burgess gave the audience a more complete description of the nature of the programs at each university, which, although developed separately, are remarkably similar in scope and direction. The Amsterdam University program, which is compressed into one year, speaks volumes about the ability to teach system and network administration as a core academic discipline. Oslo University College offers a two-year program divided into four semesters. The first semester is spent giving students background knowledge with courses such as networking, firewalls, info security, and system administration fundamentals. The second semester teaches students to stand on their own two feet, with a course heavy on lab work, a course on how to read research papers, and a course on ethics. The third semester attempts to make students think critically about what they've learned and adds some specialized courses. The last semester culminates in a thesis that draws on the foundation of the previous coursework.

The subject of university education is very different from self-learning or even targeted training (such as that provided at LISA). Accredited academic courses immerse a student in a common culture, not only granting knowledge about the world but also teaching the processes required for abstraction and the development of generalized theoretical frameworks out of specific empirical evidence. This common culture aids in the understanding and advancement of most subjects, and judging from the success of the two programs detailed in the paper (with three groups of students from each program having gone on to professional IT positions in various organizations), system administration is no exception.

■ ***On Designing and Deploying Internet-Scale Services***

James Hamilton, Windows Live Services Platform

In this presentation, James Hamilton gave the audience a whirlwind tour of the Microsoft Live Platform and how he and his team, driven by the past 20 years of experience, developed best practices for building "operations-friendly" services. Three key tenets empower Hamilton's practices: Expect failures—try hard to handle them gracefully; keep things simple, since complexity breeds problems; and automate everything, because automated processes are testable, fixable, and ultimately much more reliable. Another strong guiding belief is that 80% or more of operations issues (in Internet-scale services) originate in design and development, primarily of applications. As a consequence, if one wants low-cost administration, one must abandon the long-held view that there must be a firm separation among development, test, and operations.

What tasks do operations folk perform in Internet-scale service environments? Because the services change frequently, 31% of their time is spent deploying new applications and features, and 20% entails incident management for problems with known resolutions. According to Hamilton, if done right, both of these are eminently automatable. The most important reason for automating simple incident management is not the relatively small cost of personnel; rather, it is the fact that the more frequently a human being touches software or hardware, the greater the chances of breaking something.

When automated, these tasks may improve the operations-friendliness of your infrastructure by a factor of 2. But Hamilton takes things to the limit with a tenfold increase via recovery-oriented computing (ROC), better designs for applications, automatic management and provisioning, incremental release, graceful degradation, and admission control. ROC assumes that software and hardware will fail frequently and unpredictably. Applications and servers should be heavily instrumented to detect failures. Different failures are caused by two different types of bugs: Bohr bugs cause repeatable functional failures and were usually generated in development. Monitoring should report these with high urgency. Eisenbugs usually occur because of a confluence of ill-timed software and hardware events. Recovering from them involves a series of steps such as rebooting, re-imaging, and finally replacing offending machines.

On the application side, there are some best practices to follow: Develop and test in a full environment; continually perform service health checks in production; make services run on fault-isolated clusters of servers; implement and test the tools that operations will use as part of the service; and partition and version everything. The principles for auto-management and provisioning include the following: Expect to run services over geographically distributed data centers, even if you don't do it now (making your service resilient to high latency); manage "service roles" as opposed to servers (i.e., develop one image, test on that image, and install that image on identical servers); force-fail all services and components regularly, when people are around to fix them (i.e., if you don't test an execution path, expect it not to work); and most importantly, make certain that rollback is supported and tested before deploying any applications. Rollback works if you always use an incremental process with two or more phases. Finally, when capacity planning, remember that no amount of "head room" is sufficient. Unimaginable spikes will always occur. Instead of wasting resources, find less resource-intensive modes to provide degraded services. If you're ultimately backed into a corner, Hamilton gives you permission to practice admission control (e.g., drop requests from new users).

- ***RepuScore: Collaborative Reputation Management Framework for Email Infrastructure***

  Gautam Singaraju and Brent ByungHoon Kang, University of North Carolina at Charlotte

No one who uses a computer needs to hear another story about how bad the email spam crisis has become. But did you know that NACHA (an electronic payment association) estimated 2004 losses to phishing alone at US $500 million? Email providers and researchers have been fighting spam in various ways for a long time, through content-based filtering, real-time blacklists (RBL), PGP, bandwidth throttling, sender authentication (DKIM, SenderID, SPF, etc.), certification schemes (e.g., Habeas and SenderPath), and reputation management (Gmail). All of these handle spam to different degrees and organizations tend to employ more than one technology to keep ahead of spammers. But why should organizations (especially those with a small user base) fight this menace in relative isolation? Why not band together to leverage their various chosen technologies as a powerful antidote to spam? RepuScore proposes to aid this collaboration.

RepuScore, an open-source effort, allows organizations to establish the accountability of sending organizations based on that sender's past actions. It can be deployed alongside any existing sender authentication technique and collects reputation votes (in favor of or against senders) from existing spam classification mechanisms and individual users. Each organization computes its own reputation "view" of the world and submits it to a central RepuScore authority, which in turn continually generates global sender reputations (i.e., how much can I trust this sender?). The architecture is hierarchical: Each participating domain maintains one or more RepuServers, which classify senders via filters and compute local history-weighted reputation scores for each peer. These statistics are aggregated in a single RepuCollector that averages reports from every server. A single vote on reputations is then sent to a central RepuScore authority, which implements a weighted moving average continuous algorithm.

The original RepuServer algorithm works as follows: In each interval, a current reputation is computed for every sender domain (domains from which email was received) as CurRep = (# of good emails)/(total # of emails). The reported reputation is then calculated as *alpha* * ReportedRep(previous interval) + (1 - *alpha*)*CurRep, where *alpha* is a correlation factor that essentially determines the importance placed on the past reputation of the sender. A lower *alpha* emphasizes current reputation over past reputation, and vice versa.

Singaraju remarks that the ideal behavior for a reputation management system is to have a slow increase in reputation as an organization "proves" itself, but a quick decrease in reputation if an organization starts behaving badly. To achieve this, the researchers modified the original algorithm to only increase reputation if the sender improved from one interval to the next, and to decrease reputation if the sender did worse than in the previous interval. In various tests, high values of *alpha* were able to achieve the desired behavior while remaining resilient to various attacks (e.g., Sybil attacks, which weaken reputation systems by creating a large number of pseudo-identities). RepuScore does make some assumptions—for example, that all RepuServers at your location are secure and reporting correct information and that many organizations are participating—in order to deliver an effective solution.

INVITED TALK

- ***Homeless Vikings: BGP Prefix Hijacking and the Spam Wars***

  David Josephsen, Senior Systems Engineer, DBG, Inc.

  Summarized by Tung Tran (tunghack@qmail.com)

Dave said that in the history of spam wars, there are two primary categories of defense: IP-based and content-based spam filters. Dave asked the question, "Who is the biggest user of SPF?" The answer: spammers. (The audience member providing this correct answer received a free book from Dave.)

He then explained BGP prefix hijacking (prefix hijacks make the IPs of others your own) and gave an example to show how it works. Moreover, he pointed out the fundamental reason for BGP's vulnerability: BGP is designed for cooperative use. He showed us how to be a spammer: Get a T1 connection or be a shady ISP.

The Q&A was very intense, with the main discussion focusing on IP-based and content-based spam filters. The first questioner disagreed with the speaker's theory about the attack (BGP prefix hijacking). Dave admitted that the BGP attack might not be too popular. Some questioners raised the issue that content-based spam filtering is not scalable and IP-based filtering is cheaper and still works. They supported their argument by mentioning their specific issue: receiving more than 1 million messages a day. However, the speaker and some others disagreed with this idea. They said that the problem lies not with the content-based filter, but with the implementation of this method. They also asked those who support the IP-based method to publish a paper to better outline their idea.

INVITED TALK

■ *Beyond NAC: What's Your Next Step?*

Mark "Simple Nomad" Loveless, Security Architect, Vernier Networks, Inc.

Summarized by Nathaniel Husted (nhusted@iupui.edu)

Mark said that Network Access Control (NAC) provides a way of regulating and controlling access to the network. A NAC initiates this process when the machine starts up and tries to access the network. NACs are also a way of enforcing policy on the endpoints of the network. Mark specifically stated they are neither a security nor policy solution but an enhancement.

Mark also discussed how NACs should be implemented and how they should perform. He illustrated this by telling a story: A VP is at an airport, trying to make a very large business deal that is worth hundreds of thousands or even millions of dollars. The VP has visited various Web sites, thereby infecting his computer with malware, viruses, and other programs of ill repute. The question posed at the end of this story was, Do you allow him onto your network to look up some information and close the deal? Mark said that ideally you should, but make sure he only has access to the resources he needs to complete the deal. This will mitigate much of the damage he could do if he had full network access.

The NAC should also be an inline solution and it should be very fast. The latency should be under 1 millisecond and should be in the microsecond range. This includes any IDS or IPS services involved in the solution. The solution must react to events in real time. It also must be able to react to a very large number of these events. It must work as well with thousands of users as it does with one user. The IDS and IPS services the NAC implements must also be state of the art. They must be able to decompress GZIP traffic on the fly and handle the numerous protocols that are out there. The NAC should also be seamless and scalable. It should require no changes to the existing infrastructure, regardless of the size of the infrastructure.

Another problem with implementing a NAC is deciding upon ownership. Mark said that one customer actually did not purchase a NAC solution because the customer could not decide what department within the organization would have control of the NAC. Since so many different departments are involved in a NAC solution, choosing who runs it is an integral part of its deployment.

NACs are also not a static solution; they need to adapt. Attackers are constantly finding ways to bypass NAC solutions. These attackers were not botnets or external sources, but contractors and evil end users. Many users will do this just to avoid NAC policies. This can be done by spoofing various data. This data spoofing generally involves default policies applying to equipment such as printers. Also, certain MAC addresses that are allowed to manage items on the network are prime spoof targets. NACs have to adapt not only to attacks but also to new technology. They have to support all platforms and changes to those platforms. They also need to adapt to policy changes (e.g., a directive forbidding management accounts from being allowed on local computers). NACs also need to enact policy post-authentication and be able to cover a broad range of policy decisions, such as allowing IM but disallowing file transfer within IM. Mark suggested that the IPS be tied into the system and able to enforce these policy changes on the fly.

Another important factor of a NAC is its ability to cope with the ever-increasing mobile workforce. The NAC must work over wireless, dial-in, and VPN interfaces. The NAC also must deal with contractors and guests who require access at your organization. Mark suggested that all NACs should at least have the ability to access the Internet and use a single printer.

NACs are not a replacement for perimeter technology. Antivirus servers, firewalls, and other network security devices are still needed to protect against client-side attacks. You should also have technology in place to protect against alternate routes of attack.

Mark also discussed some things that NAC vendors do not tell you. The first is that after authentication, users still could be doing bad things. It is possible for the user to spoof information that checks for system compliance so that their virus-laden computer can connect to the Internet. Mark again stressed the fact that, to be effective, the NAC solution must be inline, in the core, in the perimeter, and everywhere else on the network. The system must be a mediator among all users on the network, and some NAC solutions are not. Vendors also will not tell you that NACs only control access to network resources and do not control access to applications and data independent of network resources. Mark also said that tunneling protocols can bypass virtually all vendors on the market. Also, NACs do not help if sensitive material in need of protection resides in the data that can be accessed. One example Mark gave involved a person with legitimate data access collecting data snippets and combining them to form a position for insider trading.

Mark finished his talk by discussing where NACs are heading. He sees future NACs being able to identify more than just who is accessing the network; they will also identify what data they are using and what applications they are using to access that data. Future NACs will be able to use layered profiles to limit network access. They will limit access based on user identity, application usage, and data usage. He also sees NACs providing easy correlation of events to help administrators put seemingly unconnected events together to solve a bigger security puzzle. Mark also sees NACs providing more automated reactions to events on the network than current IPS solutions. In general, he sees NACs becoming more automated as time progresses.

There were few questions asked during this session but the answers provided were detailed. Mark was asked his opinion of signature-based IDS and IPS systems and how viable he thought they were in the immediate future. Mark said that he wasn't a big fan. He suggested that a company have at least one commercial solution but should back it up with Snort. He also suggested that a company should use a combination of both anomaly- and signature-based systems to cover the full range of scenarios.

The second question concerned agents and the effectiveness of NACs against encrypted traffic such as SSH. Mark explained that an agent would ideally be in programmed Java to allow for maximum cross-platform usage. In general the NAC will be unable to read SSH traffic, but in some regards it can be predictable. He said that some studies have shown that the first thing an administrator does when logging into a machine is type `su`. This has a clearly defined length and could be detected even if the traffic is encrypted. There is nothing more that can be analyzed from the encrypted traffic beyond correlation.

The final question was about the validity of Gumjack (systems on a USB device) in NAC situations. Mark said that it could scale well but you would have to buy a Gumjack device for every employee's computer, so the logistics and economics could be daunting.

INVITED TALK

- ***The Economic Meltdown of Moore's Law and the Green Data Center***

  Kenneth G. Brill, Executive Director, Uptime Institute

  Summarized by Kimberly McGuire (klmcguir@iupui.edu)

According to Moore's Law, the number of transistors on a piece of silicon will double every 24 months. In fact, the number of transistors on a piece of silicon has been doubling every 18 months, faster than originally predicted by Moore's Law. However, this increasing rate of computational performance is greater than the rate of power efficiency improvement. The result of multiplying increasing computational performance with energy efficiency improvement is that more and more electricity is being consumed at the plug. In 2005, Dr. Koomey of Stanford and the Uptime Institute estimates that servers used 1.2% of the electricity generated in the United States; this figure is up from 0.6% in 2000.

This lag in power efficiency will drive a site's Total Cost of Ownership (TCO) up and reduce economic productivity. Because of these increasing power needs, square feet costs of a data center are irrelevant; costs will be driven by the power consumption of the IT equipment. The increasing demand and cost of electricity has big tech companies moving to areas where they can get power at less than $0.03 a kilowatt-hour or to areas of the country that have surplus power.

Mr. Brill suggested four metrics to determine whether your data center is "green": (1) IT strategy optimization; (2) hardware asset utilization; (3) energy-efficient hardware deployment; (4) site infrastructure overhead minimization. Turn off machines that aren't doing work, virtualize, and use what you have efficiently. As spindle speed doubles, power consumption goes up by a factor of 8. Does everything need to be on the fastest disk? Buy energy-efficient hardware. It's available but currently does cost slightly more, but you'll likely save in incremental cost. Remember that less than half of what comes out of the plug goes to computation. The remainder is overhead.

A green data center for a large global enterprise can make an estimated $100 million in profit or competitive advantage over 10 years. Scaled-down savings are available for smaller centers.

Business units want the latest and greatest equipment for their money. However, IT's current economic chargeback systems typically fail to take the true cost of ownership into consideration. Part of those new, faster computers is the cost of electricity, which as a single site cost element all by itself will soon exceed the cost of the server over three years. Unfortunately, this is only one of several major site infrastructure cost components that need to be billed back to users.

Until the IT cost chargeback system is fixed to determine true costs, users will be motivated to make suboptimal decisions. It is the responsibility of an IT manager to explain those true costs and try to convince business units that they don't have to sacrifice much in performance to see a substantial reduction in the three-year TCO for a piece of equipment. Ideally, chargeback to those business units is key to containing TCO for a site.

Another way a company can reduce site costs is by using the Information Technology Infrastructure Library (ITIL). ITIL was originally developed by IBM for the United Kingdom. ITIL uses checklists and detailed descriptions of processes and practices that can be tailored to fit any IT organization. Mr. Brill pointed out that a site's IT equipment needs to be included in the configuration database when building or moving to a new data center.

CONFIGURATION MANAGEMENT

Summarized by Kevin James (kevljame@cs.iupui.edu)

- ***Moobi: A Thin Server Management System Using BitTorrent***

  Chris McEniry, Sony Computer Entertainment America

Chris presented a solution used by Sony Computer Entertainment America (SCEA) to update and deploy their 2000+ game servers. Moobi is an image distribution system based on PXE, DHCP, TFTP, and BitTorrent that he developed after years of searching for ways to deploy his many server instances and minimize downtime. He began by describing the motivation for Moobi.

SCEA's online gaming servers grew from 350 in 2004 to more than 2000 in 2007, while tasking two administrators at most to maintain them. Initially he turned to Cfengine but found that he was unable to express what he needed using this powerful tool. Chris is quick to say that the problem was not with Cfengine; they simply couldn't accurately express what they needed in their classes. In one instance, after updating the ntp configuration on the game servers, several cascading events caused the servers to crash. Realizing several deficiencies in their process, Chris was inspired by the Linux Terminal Server Project (LTSP). LTSP works by intercepting the normal boot order of the kernel. He thought, "Why can't we do other environment setup here?" The decision was to load an image during this step.

Unfortunately, the booting of the servers became a bottleneck. He decided to leverage the spare capacity of the servers and available network bandwidth and run the BitTorrent client during the initial kernel boot. By sending different segments of the image to different nodes within a subnet, not only are the servers able to load an image faster but slow-to-start servers are able to quickly catch up. After running a controlled experiment, Chris reports that Moobi running with only three boot servers was able to update an impressive 600 nodes in approximately an hour, with 95% of the machines finished in the first 15 minutes. The last 5% were slowed by PXE boot failures.

In the future, Chris plans to port Moobi to more OS distributions and provide better hardware detection on boot. Another improvement would be to develop a method for integrating Moobi into existing configuration management tools.

- ***PoDIM: A Language for High-Level Configuration Management***

  Thomas Delaet and Wouter Joosen, Katholieke Universiteit Leuven, Belgium

  Awarded Best Paper!

Thomas Delaet presented PoDIM: an object-oriented language aimed at creating high-level specifications for configuration management tools. Instead of defining what processes are necessary to complete the configuration of a host or how to enforce the configuration, PoDIM focuses on defining the relationships between different entities within the host and network, delegating the details to various tool-specific translators. He cites Paul Anderson's paper "Towards a High-Level Machine Configuration System" given at the 8th annual LISA Conference [*Proceedings of the 8th Large Installation Systems Administration (LISA) Conference* (Berkeley, CA: USENIX, 1994), pp. 19–26] as a reference.

In PoDIM, "all 'things' are objects." It leverages existing research in language creation and software engineering in the "Rule Language," used to define your site policy. Delaet lists static typing, multiple inheritance, and contractual programming constraints (preconditions, post-conditions, and class invariants) as advantages of their approach. Another advantage is the use of object references when attributes refer to other objects versus actual object copies. This is used to define dependencies between different object classes. One creates a site policy by creating several rules and constraints that define the composition of each object as well as how their attributes may be modified. This is accomplished by using an SQL-like syntax.

After the site policy is defined, it is fed to the PoDIM compiler, resulting in several complete object descriptions, much like a normal compilation step. In response to an audience question, Delaet explained that failures during this step do not necessarily cause the entire compilation process to fail. Only objects that depend on a particular rule definition, either directly or through reference, fail during compilation. These are used by a configuration-tool-specific templating engine, which generates the files necessary for that tool. The code that results is then supplied to the configuration system. This allows for easy integration into current configuration frameworks. The current reference implementation for the templating engine generates Cfengine code.

In the future, Delaet plans to introduce greater modularization into the translation process by separating the rule logic from the object definitions. Other improvements include simplifying the integration of PoDIM into higher-level tools and GUIs, creation of templating engines for other tools (LCFG, Bcfg2, Puppet), as well as a method for translating the native configurations of such tools into PoDIM. Finally, he stressed the need for a communication mechanism to facilitate the resolution of cross-machine dependencies.

- ***Network Patterns in Cfengine and Scalable Data Aggregation***

  Mark Burgess and Matt Disney, Oslo University College; Rolf Stadler, KTH Royal Institute of Technology, Stockholm

Matt Disney presented the results of work toward introducing decentralization into Cfengine. Recognizing that centralized management strategies will eventually fail on some level, they took cues from network management patterns to develop decentralization schemes in Cfengine's monitoring. Drawing on graph and tree traversal algorithms, they developed a logical overlay network for each scheme, independent of the actual physical layout of the network. Each scheme is characterized by an expansion phase, during which nodes are queried, and a contraction phase, during which the responses of each node are aggregated. One highlighted application of this approach is in the field of autonomics, during which such feedback from nodes is necessary in the reconfiguration process.

To study the behavior of these schemes, three different experiments were developed, and statistics were collected over 50 runs. They first tested a scheme called Echo, in which a query is pushed to nodes arranged in a tree overlay and then the responses are collected; this is repeated and compared to the performance over parallel star and serial star overlays. The results from this experiment showed that although the parallel star overlay performed the fastest and the serial star overlay required the lowest workload, the echo overlay provided a nice trade-off between the two, running in half the time of the serial star while generating one-fifth the workload of the parallel star.

The next experiment, GAP chain, arranged the nodes into a chained overlay, but did not use an expansion phase. Instead, responses were simply aggregated from the nodes. After test runs similar to the Echo experiment, the results showed that the GAP chain performed even better than they had originally expected. Further investigation of this showed that adding a sleeping factor to the nodes increased performance, even allowing for a single-cycle update after some adjustment.

The third experiment used the same methodology as the second, but with a tree overlay. The results showed the performance of this overlay to be quite stable, but similar to the chain; greater performance can be achieved by adjusting the sleeping factor attached to the nodes.

Although the results of their experiments are quite promising, Disney does report some limitations and errors they encountered. Virtual machines were used to simulate the test network; therefore the time on each node could have become skewed. Also, the sample sizes were small, only 20 nodes; he believes that more representative results could be obtained with larger sample sizes. In conclusion, they plan to implement more pattern overlays in Cfengine. To facilitate this, the group plans to explore enhancements to pattern specification in Cfengine.

INVITED TALK

■ *Hardening Your Systems Against Litigation*

Alexander Muentz, Esq.

Summarized by Kimberly McGuire (klmcguir@iupui.edu)

First and foremost, Mr. Muentz does not work for Microsoft. The information contained in this summary or in his presentation is not legal advice and is for informational purposes only. This area of the law is in flux and what may be law today may not be tomorrow.

Civil litigation is an IT risk for which preparations must be made in the event a lawsuit is filed against your company. Civil litigation poses a security risk as it allows outsiders to view and handle sensitive data and could potentially lead to financial losses for you and/or your organization. There are myriad reasons a person or persons could file a civil suit against a company.

A civil lawsuit starts with a complaint that lists all legally supported claims. The next step in the process is discovery. During the discovery process each side produces all responsive information related to the lawsuit. Additionally, during discovery each side gets to interview, under oath, selected individuals from the other side and each side can subpoena information from third parties with relevant information. Discovery is based on good faith; if either side fails to produce relevant information purposely or accidentally, they can face fines, data recovery fees, dismissal of claim or defense, dismissal of lawsuit, or loss of suit. Finally there is a settlement, trial, or arbitration to determine the outcome of the lawsuit.

Litigation is so expensive primarily because of the discovery process. Once a civil lawsuit is filed a litigation hold is put into place requiring you to preserve all responsive data and documents. Data and documents include but are not limited to email, digital documents, voicemail, backup tapes, system logs, and slack space on disk drives. Then the data and documents are collected and a discovery conference is held. During this conference each side discusses the sources and people they have and sets a schedule and format. After the discovery conference all the data is reviewed at least twice—in some cases, three times. The first review is usually done by a junior attorney; the second and third reviews are done by more experienced lawyers. At between $90 and $150 an hour for each lawyer it is easy to see how quickly the expenses can grow.

The discovery process is also the biggest security and privacy risk for a company and its employees. It is a privacy risk for employees because of the grey area between personal lives and business. It is no longer uncommon to find people who work from home on a regular basis or use personal email for business. It is an IT security risk because of the broad sweep of the process. The law firm takes everything and anything that may be related to the civil suit. The law firm may have inadequate security or may contract out some of the work to third-party vendors, leaving sensitive data in insecure hands.

What can you do to prepare yourself? Do an ESI (Electronically Stored Information) audit. Identify all key systems and determine their contents. Use policies to define retention of ESI, how users can remotely access systems, the decommissioning of systems, and use of personal email for work, and follow those policies. Finally, implement a collection plan for end-user PCs and file servers. Preparation is key.

There are also some steps you can take if you find yourself already in the middle of litigation. First and most importantly, cooperate with your lawyers. Enforce the litigation hold and request additional storage capacity to handle the additional data. Attend the discovery conference, assist in working out a technical plan, and be prepared to correct any bad technical information the other side may be trying to pass off as legitimate. If required, help select third-party vendors to ensure that data is reviewed in a secure location. If you are deposed, explain exactly what you did and why you did it.

Alexander Muentz is based in Philadelphia, PA, and is licensed to practice in the state courts of New Jersey and Pennsylvania. The slides from his presentation are up at http://www.usenix.org/events/lisa07/tech/muentz_talk.pdf, and a related article appeared in *;login:*, Oct. 2007.

SPECIAL LUNCH & LEARN

Should the Root Prompt Require a Road Test?

Alva L. Couch, Associate Professor of Computer Science, Tufts University

Summarized by Kevin James (kevljame@cs.iupui.edu)

At this year's Lunch & Learn session, Professor Alva Couch led a discussion on an issue that has great importance for the future of the system administration profession: What makes a good system administrator and how do we measure this? Alva began by asking, "Is there a mysterious compound 'W' that makes system administrators functional and ensures success"? Often certification is thought of as being the proper way to become a good system administrator, but Alva believes that we have this relationship backward: certifications serve experienced system administrators far more than new ones. He takes the stance that certification tests cannot measure what makes a good system administrator, "Quality X," but instead measure an individual's knowledge of a specific product or brand. Yes, we should attempt to determine functional system administrators by certifying them, but there are some things that tests cannot measure.

He proffered driving as a metaphor for our current problem, as "we are drivers of the technological revolution." We certify a driver's knowledge using a written test and skills using a road test; our current testing frameworks accomplish the first well enough, but there isn't an equivalent for the road test in system administration. Again, Alva believes we are asking the wrong question. A better question would be, "What is the difference between new drivers (and sysadmins) and more experienced ones?" Accidents: New drivers are associated with higher accident rates, and rates seem to decrease with the experience of the driver. Alva credits this to an increase in situational awareness and judgment, which allows drivers, system administrators, and even pilots to "understand the broader effects of your actions." These make up our Quality X, that which makes sysadmins good. This quality is only attained through causing accidents and learning from them.

Having identified Quality X, Alva continued breaking down its role in system administration. Situational awareness entails determining not only what could be wrong and what could have caused it but also the side-effects of your solution both on your systems and on your consumers. The extent of one's situational awareness can be considered the "maturity level of a system administrator"; when solving problems, whether our focus is limited to the local system or extends to the whole enterprise and beyond to your lifecycle planning depends on the amount of experience you have attained. Mentoring becomes indispensable; the inexperienced can learn from their mistakes in an environment where someone can guide them on a path to greater awareness.

He concluded that we will never be able to measure maturity as an administrator and therefore should give up on knowledge-based tests to achieve this. Instead, we should focus on increasing the availability of mentoring and other methods for expanding experience. Handing the discussion over to the audience, Alva left us with a few thoughts. In the technological revolution, we are drivers. We want "professional" status and "respect." Please drive safely.

WORK-IN-PROGRESS REPORTS (WIPS)

Summarized by Gautam Singaraju (gautam@singaraju.com)

Fettle, a Program for Populating and Dressing Racks

Andrew Hume, AT&T Labs—Research

Maintaining multiple numbers of servers and racks poses a significant administration problem when placement at different locations is required. Andrew Hume developed a program that allows users to place the servers on racks based on the number of Ethernet connections, power supplies, switches, etc. The tool creates a 3D presentation to show how the servers can be placed. The tool, Fettle, should be available soon on sourceforge.net under an open source license.

Excellent Performance with Aging Hardware

Alberto D'Ambrosio, National Institute of Nuclear Physics, Turin, Italy

Citing the Brooklyn Bridge as an example, Alberto D'Ambrosio suggested that system administrators now have to monitor and support systems developed by others. At his organization, two machines were used as mail servers. These could handle the load for the first few years, simply fixing any problems that started showing up. However, as spam started increasing, the servers began to reject emails, and processing and storage increased tenfold. The cluster had become less reactive owing to SCSI starvation. Performance increased once they relocated to a Bayesian database. They started recycling their old servers to provide additional performance benefits.

■ ***What's New in Amanda: The Open Source Backup Platform***

Dustin J. Mitchell, Storage Software Engineer, Zmanda, Inc.

Amanda supports a device API that allows pluggable storage backends such as tape, disk (vtape), RAIT (Redundant Array of Independent Tapes), WAN, and optical media. Application API integrates Amanda with tar, dump/restore, different databases, Windows, AFS, and NDMP. The Amanda transfer architecture (XFA) has a client-server model, in which the client passes the messages to the supervisor, which is present on the server over the network. The client compresses the data received from the application, which is then encrypted by the server and sent to the taper. Perl in the code allows new contributors to join the system while providing low-level processing and using the high-level libraries. Dustin Mitchell invited new developers and contributors to join in development of Amanda.

■ ***Analysis and Verification of XACML Policies***

Saurabh Arora, Pablo Giambiagi, and Olav Bandmann, Security, Policy and Trust Laboratory, Swedish Institute of Computer Sciences, Sweden

XACML provides access to a rich language for expressing security policies and makes it possible to integrate many different authorization models into the same framework. Policy management tools need to be enhanced in order to help system administrators design sound policies, support policy change management including policy optimization, facilitate cooperation among administrators, support GUI functionality, and support properties that are not directly expressible in standard policy languages (e.g., Separation of Duties or Chinese wall). In the upcoming XACML 3.0, one of the most important additions is a mechanism for delegation of security administration. It provides a rich language for expressing security policies in which policies can be issued by authorized issuers. The new specification can be used to implement decentralized administration in applications and a mechanism for delegating security administration.

The authors developed a Policy Analysis Subsystem (PAS) that translates policies and queries to propositional logic. External data such as XACML policies, attributes, and relations can be fetched by PAS to compose queries. Saurabh discussed SAT solver as a tool for (a) solving the Boolean satisfiability problem and (b) analysis of counter-model examples. PAS can iterate queries and/or adapt queries based on results of previous queries, as needed, to express higher-level queries.

■ ***Grid Services at Yahoo! Comes to LISA***

Marco Nocosia

Marco Nocosia introduced Hadoop, a distributed file system and map-reduction programming platform designed to scale out to satisfy the requirements of a full-scale Web content system. Typical SAN and NAS do not support enough storage or IO bandwidth. Hadoop combines the storage and computation power of any set of computers and is written in Java; the user's program can be written in the user's preferred language. Hadoop has been developed as an open source project and interacts with HDFS. The information can be seen in a browser and Hadoop allows the clusters to run both DFS and M-R.

■ ***MachDB***

Nathan Hubbard

Every organization rolls their own machine DB, which is usually independent of the organization's. MachDB is a scalable, open source implementation of the most needed of system administrator tools to maintain the machine database. MachDB began as a quickly growing startup, but it is scalable for enterprise environments. There are several design goals for MachDB: Information gathering should be architecture- and OS-agnostic, the back end should be LAMP, the XML spec should be the API, and it should be scalable to 10,000+ hosts, have human and machine readable interfaces, have an easy-to-use Web front end, and offer a history on everything and templates for easy UI modifications. The code is now in its alpha phase and will be released in a few weeks at the project Web site: http://www.machdb.org.

■ ***mfork***

Maarten Thibaut

Maarten Thibaut needed to synchronize huge amounts of data among multiple servers. Serial rsync does not serve this purpose because it requires too much time. Maarten suggests using parallel rsync, which uses a fork mechanism called mfork, a simple command that allows parallelization. The command mfork forks rsync and copies data. Make was not used, as it depends on gnu and takes additional time. Parallel rsync also allows users to save the results without any issues of parallelization.

■ ***Migrating to Internet Protocol Version 6 (PDF)***

Dennis Underwood and Jon Lavender, University of North Carolina at Charlotte

Migration to IPV6 is a long-term necessity, but experience with the protocol is limited and usage and implementation policies need to be established. However, migration from IPv4 to IPv6 affects the entire network and middleware will need to be replaced with end-to-end administration policy. Although the new protocol eases network administration, associated technologies keep developing rapidly. Dennis Underwood and Jon Lavender suggest that policy should be valued first and different alternatives should be devised for short-term and long-term migration strategies. Ignoring IPv6 is not an option; their alternate option of immediate migration has advantages and disadvantages. Because the models are still developing and vendors may not leverage all IPv6 capabilities, immediate adoption is simply not possible. The authors suggest the development of long-term policy strategies to gain eventual full IPv6 connectivity.

- *User Satisfaction and Technology Acceptance Among System Administrators*

  Nicole F. Velasquez, University of Arizona

Nicole Velasquez studies user satisfaction and technology acceptance among system administrators. Human-Computer Interaction (HCI) among system administrators is completely different from that of the usual users. Presently, more money is being spent on human development cost. Nicole uses verification and login information to determine information quality. The system quality that is being used should be reliable, flexible, and integrable. Nicole proposes scalability, credibility, and situational awareness.

- *Frequency Domain Analysis and Visualization of Web Server Performance*

  Marc Chiarini and Alva Couch, Tufts University

Frequency domain analysis and visualization of the Web server provide an external model of behavior of the Web server. It allows one to check how a Web server responds to simple requirements. With the help of frequency analysis, the server measures the effects of increasing load and checks for abnormal behavior. Input classes are important for performing frequency domain analysis because they take into account the different types of inputs coming into the system by checking the caching mechanisms, dynamic pages, and database server performance. The authors demonstrated their frequency domain analysis using graphs and discussed how the frequency domain analysis helps in plotting expected domain behavior and the performance of the servers.

- *How Do You Protect the Data of 1500 Very Nontechnical People?*

  Ski Kacoroski

Dealing with a nontechnological user base usually implies that people cannot back up their critical information. The solution presented here is to use backup.sourceforge.net. This enabled machines at a K–12 school to be fully backed up. The helpdesk has a Web site that can be used and backups can be made using different techniques. The system is optimized to back up the users' data at night.

INVITED TALK

- *Prince Caspian on Location: Commodity Hardware and Gaffer Tape*

  Trey Darley, Technical Consultant

  Summarized by Kimberly McGuire (klmcguir@iupui.edu)

Working on the set of a big-budget, major motion picture sounds like a great job. Traveling across Europe to exotic locations, Trey Darling did this for the new Narnia movie, *Prince Caspian*, for approximately six months. Trey worked with a team of four for Walt Disney Productions and Walden Media. He started working at the Barrandov studio in Prague, Czechoslovakia. He immediately found himself working as a reactive system administrator. Because of the way contracts were set up, cast and crew brought their own equipment, including their own computers. This myriad of computers required expertise to support OS 9, Vista, and everything in between.

Working in a building constructed during World War II presented its own problems. Thick walls made it difficult to run cable. Requests for connectivity would come in on a Friday to be completed by Monday. The original network was to be for 50 to 60 users, but by the time Trey left the project the network was handling 500 to 600 users. The Linksys switches, originally selected for a network of 50 to 60, had to be replaced every two weeks. Management determined that it was less expensive to buy and replace the cheaper switches than to spend more money on switches that could adequately handle the traffic.

Next on the location list was Bledne Skaly, Poland, a small town in the middle of nowhere. The base camp consisted of trailers and tents, both of which changed configuration daily. Trey started in a pop-up tent, with a battery and a bench. Everything in the tent, including equipment, had to be packed up during thunderstorms, which hit the area regularly. The IT department eventually got its own trailer. There was no Internet access in Bledne Skaly, so they had to use a wireless modem. The base camp and "the set" were divided by a stretch of woods. To get Internet access to the set from the base camp required a series of wireless access points to be constructed through the woods. Trey and his team used garden hose as conduit to run cable through the muddy fields that surrounded the base camp.

Good-quality services were difficult to find. An example Trey gave was when one of the stars of the movie required a data recovery service on a dropped laptop. The team found a company to do the data recovery, but the hard drive was seized by the police during a raid on the company. After "talking" to the right people the drive was recovered.

Avid Unity systems were used for redigitizing and cutting the movie. Each Avid system stored 25–30 terabytes of data. There were no backup plans in place for these systems and instead of getting good-quality power converters, each $20,000 system used a $20 converter.

Trey was asked whether he would do this kind of work again. While he never said yes or no to the question, he did say that if he did it again he would press production hard to get a better feel for the scope.

INVITED TALK

- *The Security Butterfly Effect*

  Cat Okita, Earthworks

  Summarized by Nathaniel Husted (nhusted@iupui.edu)

Cat's talk involved quite a lot of audience interaction. In fact, in a response to an audience member, she said that if you have ethical problems with raising your hand, please feel free to raise any other body part. There was also a member of the audience who provided sound effects for the technical session. Although these portions of the session cannot be recreated in the summary, the educational information can. It is recommended that the readers view her PowerPoint slides at http://www.usenix.org/events/lisa07/tech/okita_talk.pdf while reading this summary.

Cat introduced her talk by posing the question, “Does the flap of a butterfly’s wings in Brazil set off a tornado in Texas?” This was a question asked by a paper in the 1960s. The author of the paper did one set of calculations, then manually entered his calculations a second time and received a completely different answer. The Butterfly Effect is when small variations in the initial condition of a system produce large variations in the long-term behavior of the system.

Cat then defined the characteristics of the Butterfly Effect, or rather what it is not, since it is a subtle effect. It is not the domino effect, nor is it linear or cascading. It does not involve one clearly identifiable thing leading to a problem that spans out.

She then asked whether any of us know the initial conditions of our systems. The answer to this question is no. As we become more specialized and modularization increases, and as our software becomes more complex, this situation will get worse. It is from this lack of knowledge that we make assumptions about our systems.

Cat split the assumptions we make into three categories: environmental assumptions, behavioral assumptions, and blind spots. Environmental assumptions are those in which we think we know everything about our environment. As an example, she gave “Everyone knows it doesn’t snow in the deserts.” In fact, sometimes it does. Behavioral assumptions are those based on how people and systems behave. Blind spots are situations when we think a system always works in a specific way.

The rest of Cat’s talk was structured as nine specific stories created from these three assumptions. The stories were split into three sections: the cast of characters involved, the problem, and the assumptions.

The first story was about THERAC-25, a medical linear accelerator produced by Atomic Energy of Canada Ltd. (AECL). It was inspired by the THERAC-6 and the THERAC-20. These older models were created by AECL as well as CGR, a French company. Before the THERAC-25 was built, there was a fallout between AECL and CGR. AECL decided to reuse the software from the THERAC-6 and THERAC-20, but not implement any of the hardware controls that the THERAC-6 and THERAC-20 used. The THERAC-25 would only use software controls, but the THERAC-25 would either underdose or overdose the patients. It would also produce strange errors and sometimes just not work. AECL explained that this was not the company’s fault; it must be user error. After the deaths of some patients and radioactive overdoses of others, AECL finally agreed to add hardware controls. The assumptions made in this situation included the idea that a hardware company could write software and that the users were not providing accurate information. The first was a blind spot; the second was an environmental assumption.

The second story, “Samy Is My Hero,” involved the MySpace social networking site and a boy named Samy who was interested in looking at pictures of the fairer sex. To look at these pictures he needed a large number of “friends.” To reach this goal he put a little snippet of code in his profile that automatically added any viewers of his profile to his friends list. This code would then add a copy of itself to the visitor’s profile as well. MySpace was not amused. In this situation MySpace assumed that users would not and could not put malicious code in their own profiles. This was both a behavioral assumption and a blind spot.

In the third story, “How to Fit an Elephant Into a Teacup,” a secure data center was experiencing impressive growth and had a set of “world-class” access controls to secure the building. The operation of these world-class access controls was based on the weight of the subject. The problem was that the access controls thought one of the data center’s talented employees was more than one person. This happened because the weight cutoff for the access controls was 400 pounds. This also meant that multiple people with a cumulative weight of less than 400 pounds could get in. To solve this problem, the staff members propped open the fire door of the “secure” data center. There were multiple behavioral assumptions in this situation. The first was that everyone weighs the same. The second was that multiple small people would not enter the facility at once. The third assumption was that people would never go through the fire exit unless it was necessary. The final assumption was that the physical security staff would not open the doors for someone who shouldn’t be there.

The fourth story, “A Scandal in Bohemia,” dealt with paranoid privacy enthusiasts and a “security researcher” named Dan Egerstad. The privacy enthusiasts were using Tor as a way to keep their browsing habits secret. Tor is a program that redirects Internet traffic to make it hard to identify where the traffic originated. It also makes it hard to block the traffic. The fine print, however, states that it does not protect any information in the traffic and provides no guarantees. Dan Egerstad posted details of sensitive email accounts he received from sniffing the traffic at the edge of the Tor network. The paranoid privacy enthusiasts were not amused. One of the assumptions the paranoid privacy enthusiasts had was that Tor would hide sensitive information. This was a blind spot. The Tor group also made an assumption that people read fine print. That was a behavioral assumption.

The final story, “Monkey Business,” involved, well, monkeys. This story begins with the crashing of a VAX mainframe. A Digital Field Service Engineer (DFSE) was called in to fix the problem. The system administrators didn't call the researchers, because nothing looked unusual about this VAX. Problems arose when the DFSE ran a diagnostic on the system. It turns out that everyone neglected to heed a sign saying "Do not disable read-only mode" on the drive controller. After the diagnostics everyone found out there were monkeys attached to the VAX and monkeys do not operate in write mode. Some monkeys were recovered, but others experienced "fatal errors." The biggest assumption in this story was that the system administrators knew exactly what the machine was being used for. This was a big blind spot.

After Cat told her nine stories, five retold here, she discussed what she saw in the future of computing. In summary, she found the Butterfly Effect to become more prominent. Cat warned that as we continue to become more specialized people will know less and less about things outside their area of specialization. We're also experiencing a "rise of the machines." It has become more common to outsource what we know to machines and hope they properly take care of all the blind spots and other assumptions.

In response to a question about applying the wisdom from this talk to smaller systems, Cat affirmed that this Butterfly Effect analysis is most certainly applicable to smaller systems. She said that many times things will work fine with one or two programs, but when three or more are introduced, weird things can happen. In response to another question she advised that we should always expect the unexpected and we can glean wisdom from the superstition, myth, and lore we pass around as system administrators. Finally, she said that there is value in differing opinions.

CLOSING SESSION

■ ***Cookin' at the Keyboard***

David N. Blank-Edelman, Northeastern University CCIS; Lee Damon, University of Washington

Summarized by Josh Simon (jss@clock.org)

The conference's closing session began with the usual close-of-conference announcements, then we segued into "Cookin' at the Keyboard" with David Blank-Edelman and Lee Damon. While neither has any formal culinary experience, both like to cook. Lee demonstrated by preparing (from chopping the vegetables through serving it to David), on-stage while David spoke, a tofu and vegetable stir-fry and ice cream with homemade hot chocolate sauce. Unfortunately, for liability reasons, David and Lee were unable to share the stir-fry or the chocolate sauce. David spoke about how system administrators could learn from restaurant cooking procedures. First, as an appetizer, David spoke about why cooking is hard: You're not just applying heat to some food, you're managing the conditions with lots of variables, such as the quality of the ingredients, the temperature and humidity of the air, and the level of heat involved.

As the first course, David discussed recipes. Based on discussions with cookbook authors and chefs, he talked about how writing recipes is hard. You never make the same food twice, you can't go into explicit detail at every step (including such things as the suppliers of the food and manufacturers of the stove and pans and so forth) without scaring your audience, and most people don't use common sense in terms of recovery (if a recipe says "cook 10 minutes" they'll cook it for 10 minutes even if it's obviously done after 5). Solutions to these problems are to treat recipes as general guidelines and to never expect someone to duplicate a recipe but instead to approximate it. You also find that cookbooks specify common units and time ranges and provide visual and textual clues to let the reader (cook) make judgments. As you get more experience, you can come up with simpler recipes with fewer ingredients to achieve the same or better flavors. In other words, the better you get, the simpler it gets. So learn to simplify: It takes experience. Learn where to cut corners, when to ask questions, when to question every ingredient, and how to compromise when necessary.

As the second course, David had talked to several chefs about working in a world-class kitchen. Starting with a definition of restaurant terms and continuing through a comparison of trade (such as a burger-flipper or help desk worker) to craft (cook or system administrator) to art (chef or ubergeek), he went through the skills you need. The chefs agreed that to be a good cook you'd need a sense of urgency, the ability to take direction and clean up as you go, precision, a thorough knowledge of the subject matter, initiative, focus, and dedication. You also need to be part of a team, be willing to jump in and help when needed, and be able to receive new information and produce with it. These skills, with minor changes from food-industry to technological terms, describe much about system administration. In the case of cooking, preparation (mise en place) is included in the process. You need to be prepared physically and mentally; you need to know where everything is and have everything at hand, ready when you are, and be as fast and as efficient as possible. As you get more experience you're able to work better under pressure, to help others, and to show your creativity.

Finally, for dessert, David provided an overview of what we as system administrators can take away from the talk. We need to write better recipes and recipe interpreters, such as configuration management tools. We need to develop our skills and moves better. We need to prepare, work clean, and focus on the task. Finally, like a line cook becoming a chef, we need to chase perfection: Take teaching opportunities but not necessarily every learning opportunity, communicate with your team, document what you do, learn more things, ask for help when you need it, and be able to roll back and start over when you have to.

■ *Fighting Spam: The State of the Art*

Chris St. Pierre, Nebraska Wesleyan University

Summarized by Chris St. Pierre (stpierre@NebrWesleyan.edu)

Although spam is far from a solved problem, most attendees at the Spam Fighting workshop appeared to consider it about 95% or more solved. The remaining 5% still poses a concern, though, as does the fact that spammers have a growing army of programmers and computers dedicated to eliminating or obsoleting the hard-earned gains that have gotten us to this point.

The issue is complicated by the fact that nowadays spam is rarely an issue that is isolated to incoming mail. Many of the attendees had concerns about outgoing spam, whether sent or forwarded by their clients. Soft topics in spam fighting, including access and acceptable use policies, amount of functionality granted to the user, and more, were also discussed.

Currently one of the biggest guns in the spammer's arsenal is the botnet, and we detected botnets and ended spam from them. Tools such as p0f, which passively detects what operating system a given connection is from, are gaining traction since they allow mail server operators to score or reject messages that come from non-server OSes. Detecting botnets on one's own network, conversely, is getting harder, since botnet authors have started using "hide-and-seek" bots that deliberately void the high-traffic fingerprint that usually allows easy identification.

We also discussed the degree of customization granted to the end user. Although most attendees thought that allowing significant user-level customization of spam filters was ideal in theory, it is often not feasible, especially for high-volume sites, and the difference in effectiveness may not be as large as one might think.

Sender Policy Framework (SPF) and Domain Keys (DKIM) were discussed and discounted as reasonable antispam measures, although they might be useful in combination with other tools. Many spammers have embraced SPF and DKIM, which has reduced their reliability, and slow uptake by major corporations has reduced it further. Even their usefulness as antiforgery devices was debated, since they break mail forwarding as it is normally done.

Greylisting, the hot new technology from the past two years, is slowly but surely waning in effectiveness. One site reported that the effectiveness of greylisting has dropped from 21% of incoming mail dropped to only 12% in the past year.

As greylisting wanes, sender verification waxes, but it is beset by major technical difficulties. The high overhead makes it difficult to justify at high-traffic sites, even with aggressive caching. More important, spammers can use sites that perform sender verification as proxies to verify their own lists of addresses or, more nefariously, as proxies to run a Denial of Service attack against a common mail server. It was generally agreed that the potential damage one could do by enabling sender verification was not worth the benefit, which was itself high-cost.

Another high-cost antispam solution is tarpitting, which is also growing in popularity to fill the void left by greylisting. Tarpitting has roughly two forms: slowing the connection or pausing it. Either way is quite expensive, since it requires an open connection to be left open. There are several promising options, though, for reducing the cost of tarpitting. Some attendees suggested creating a purpose-built tarpit device that would pause connections, consuming resources only on that machine, and then pass the connection off to a real MTA when tarpitting was complete. Others felt it was reasonable to only tarpit after some suspicious activity was detected, perhaps as an alternative to rejecting mail that's only marginally suspicious.

Looking to the future, reputation services, including the recently launched KarmaSphere, promise to be the next must-have technology for fighting spam. At the moment, the field lacks much-needed standards, but several draft RFCs describing the SIQ standard for reputation services aim to plug this hole. Centralizing reputation services should provide a significant boon to mail administrators who have heretofore been forced to make binary spam-or-not decisions on multiple blacklists, whitelists, etc., individually.

Interestingly, we found that although spam was still a major, growing problem, email viruses had virtually disappeared. Virus writers have mostly relocated to the malware sector, where the means of transmission is generally the Web, not email. One site reported detecting fewer than 20 incoming viruses in the past eight months.

In the final segment of the workshop, we turned to one attendee's very specific issue of performance in a very high-capacity environment. Many of the spam technologies we had discussed were resource-intensive, and he needed to limit resource consumption in his 10-million-mailbox environment.

The common approaches of rejecting as many messages as possible and splitting inbound and outbound mail had already been tried, but these were insufficient. Other suggested performance tweaks were to put mail processing entirely in memory or to use machines capable of high numbers of concurrent threads.

From there, we discussed using an automated firewall management tool, such as FUT or fail2ban, to block connections from repeat spammers and create a very nimble, site-specific blacklist of sorts. This would also save the overhead of accepting connections from known spammers.

Other attendees recommended using a set of high-cost MXes as a sort of honeypot; those machines would get very slow, but this would mostly inconvenience spammers. The traffic they took away from the real MXes would allow legitimate mail more resources. As an illustration, the mail to one site's primary MX was only 86% spam, whereas the higher-cost backup MX received 98% spam.

The last solution suggested was to use an ever-growing cluster of cheap appliances. With the attractive price-point of many appliances, it could be feasible to throw enormous amounts of hardware at the problem of spam filtering in a very large environment and, by using appliances, make it essentially someone else's problem. Unfortunately, the workshop attendees couldn't agree on any appliances that had worked well across the board. Every appliance or commercial service that had worked well for one attendee had invariably worked dismally for another attendee, demonstrating the extreme variability of spam by site.

■ *MicroLISA*

Robert Au

Summarized by Ski Kacoroski (kacoroski@gmail.com)

The first MicroLisa workshop was held on Sunday, November 11. This workshop is aimed at the unique problems of sites with a limited number of staff, which means that each admin has a unique skill set, cross-training is very limited, and there is no time to specialize in storage, clustering, or other technologies. The goal of the workshop was to identify the unique problems of small sites, develop best practices, and determine how to get more of the larger community to address these issues.

Sites represented included a secondary school district, a few colleges, a computational R&D center, a small ISP, and a few startups. Some sites were standalone, whereas others were part of a larger organization from which they could get some support.

Our first discussion was on emergency and vacation coverage issues. How do you provide service when you are gone at a conference or on vacation? How can you balance your private life and the demands of work? One idea is to create the illusion of a big help desk by setting up an auto-reply message. Other ideas were to push back on management to set more realistic service levels, work to make systems more reliable to avoid pages, use service contracts to push the problems onto an external source, and rotate the pager among other IT staff to screen out noncritical issues. In terms of vacation coverage, options ranged from no coverage, people checking their phones once a day, or hiring an on-call consultant.

Our second discussion covered tools we used to monitor our systems. Nagios was the most common tool used with pages being sent to cell phones. Other tools were Intermapper, OpenNMS, and home-grown scripts. People seemed to be pretty happy with the way their monitoring tools were working. Many noted that scripts sending email were also primary monitoring tools and that it wasn't so much what was in the emails, just that they received the emails (e.g., I expect three emails an hour from this machine). In other words, we learned patterns in our incoming email. A few people felt that tools such as Zenoss and Splunk had too much overhead when compared to Nagios, SEC, or emails sent from cron jobs. Small sites need very simple low-maintenance solutions.

We had a long discussion about configuration management. How do we manage configurations? How do we determine if the overhead of a configuration management tool is worth it? A few sites used Cfengine, but most sites could not justify the overhead of setting up a configuration management tool, because they were either very heterogeneous or had very rapid changes. A mix of home-grown scripts and ad hoc solutions was the most common. It was also noted that configuration management tools puts one more layer between the admin and the machine and requires additional training, which makes it difficult to implement at small sites. Decisions on when to implement a configuration management tool at a small site were based on (1) whether it would save time and money by allowing lower-skilled staff or customers to make changes rather than the system admin and (2) whether it would help in documenting the systems.

The next discussion covered how to get help, especially expert help in areas that we just do not have time to learn in depth. The key is to get some vendors you can trust and obtain the knowledge from the vendor so you can support the system. None of us had good answers to this problem. Most folks did not like to outsource entire services because they were on the hook when something went wrong with the outsourcer and they had been burnt in the past.

Funding and working with management were discussed next. How do we convince management that equipment refreshes are a good thing? Because of funding sources, some people have big budgets for capital but little budgets for labor. In this situation people can use redundant systems as a replacement for staff. Normal maintenance and equipment refreshes are typically hard to sell to management, although some groups who are part of a larger organization are able to get this done via a central administration policy. If you are at a small site, then the best bet seems to be to determine management's pain points and figure out ways to minimize the pain in return for getting some funding for critical items.

This led into a discussion on communication with the organization. We all agreed that it is important to get out with your users and see what works and what doesn't work for them. This also helps with planning for future projects. In addition, you might be able to have the users bring some pressure on management for funding critical projects.

Once you have funding, then you have to figure out what to purchase. How do you pick a vendor? How do you test that the equipment meets your needs? What can be done to assure the vendor performs as expected? The key here is to create relationships with a few trusted vendors. For larger systems, spend the time to do testing. For smaller systems, get recommendations from people you trust and just implement them.

We touched briefly on regulatory issues and, yes, they are affecting small sites. Who needs to understand regulations? Who cares or has the liability or time? One site refused to keep critical data, believing adequate protection would be impossible for it to maintain, but many sites do not have this choice. Another idea was to outsource critical data storage (e.g., use Paypal for credit card transactions). In addition to the regulations, special privacy concerns might be important (e.g., for the rich, those in witness protection programs, etc.).

The last discussion before lunch centered on asset tracking and whether we could use that data for other purposes such as IP databases. We felt that asset tracking databases were not accurate enough and had lots of exceptions, making them not particularly useful in the system admin context. What we need is an automated way to have the equipment update the asset management database. One attendee has the beginning of a lightweight system that does this. Tools used for asset tracking were spreadsheets, wikis, and an asset tracking module on a helpdesk system.

After lunch the first topic was storage. What kind do we use? What are our criteria for picking storage? DAS is the most common at small sites because of low cost and the low level of skill needed for its operation, but it does not provide management tools. What we would like is a tool that will scan a network and map out the storage, shares, and mounts (especially when a person is just starting at a job). SATA disks were deemed to be as good as SCSI for almost all applications, but there were some concerns about costs (both training and maintenance) and the idea of putting all services onto a single storage device (NAS or SAN).

The storage discussion led nicely into a discussion about backups and disaster recovery. Most folks still use tape, although some are looking at remote disk arrays as their storage grows into the 20-TB range, because tapes are too labor-intensive. People who are part of larger organizations often make use of the resources of their parent organization.

The next discussion concerned how to train a new person. If you have the time, a good way is to have the new person do an audit of all machines and services. If not, have them shadow you for a few days or weeks and then start giving them small projects to work on. This led to how we learn and the resources we use to solve our problems. The most common learning process was to poke at a system (typically in production) and hope it doesn't break, as we do not have resources for spare test systems. Occasionally people would have time and an extra machine to work on, but often that is not the case. For getting help people used Google, mail lists, IRC, Webcasts, LISA, and the LOPSA tool page. The biggest problem we all had was how to find the good information in the deluge that we face all day (e.g., which of the 17 million books on Exchange is the good one?). A dream would be a clipping service that would have some intelligence to determine which articles should be passed on to the subscribers. Perhaps we could use blog aggregators with tags to determine good content.

The discussion then led into time management. The biggest problem for all of us was getting large enough blocks of time on a regular basis for project work, because of the daily fires we have to put out. Keeping a log of what you do helps; so does a good ticket tracking system (with RT and OTRS being the most common).

How we document and plan was next on the agenda. Wikis are the most common documentation tool. The key is to document at the correct level, which means not a step-by-step procedure, but instead a summary that assumes the reader has a basic level of competence in the subject. The biggest issue with wikis was being able to easily control access to its pages. Planning is often very ad hoc, although some sites have a budget cycle (e.g., four years for a school district) or can use trend graphs to predict what new equipment will be needed.

The last discussion centered on the unique characteristics of small shops and whether there is a good way to define them. Small shops tend to have more budget limitations, which leads them to use more open source software. Small staff size leads to many other key issues, such as time management, knowledge depth, lack of specialists, and the need to rely on consultants. Many of the tools discussed at LISA are difficult to implement at small shops because of the time necessary to learn, implement, and maintain them even though they would save time once in place. We definitely need to figure out more multipliers to make better use of our skills. Ideas were to use students, interns, and consultants.

We wrapped up with the notion of creating a mail list dedicated to discussion of small site issues and to explore the idea of a MicroLisa column in a magazine or on a Web site.

■ ***Configuration: From Managing Nodes to Managing Architecture***

Mark Burgess, Oslo University College; Sanjai Narain, Telcordia Technologies, Inc.

Summarized by Anu Singh (anusingh@cs.sunysb.edu) and Mukarram Bin Tariq (mmtariq@gmail.com)

Sanjai Narain highlighted the important role configuration plays in keeping networked systems up and running, along with the general lack of formalized tools for automating configuration to assure end-to-end communication requirements. He thanked the participants for attending the workshop and sharing their views, work, and knowledge on this subject.

Sanjai also reminded the participants of the upcoming deadline for the special issue of the *Journal on Selected Areas in Communications* (*JSAC*) on configuration; the deadline is March 1, 2008.

- ***Verification and Adaptation of Network Security Policies***

  Ehab Al-Shaer, DePaul University

Ehab talked about management of security policy configuration, a complex issue because of the many rules required—which are written in 5-tuple forms and have complex semantics—the presence of distributed devices, and distributed policy interactions. This means that often not all configurations can be checked ahead of time, leading to potential security breaches. Ehab provided an overview of existing set-theoretic formalizations of intrafirewall conflicts in distributed environments and provable sets of constraints that are sufficient. He pointed to some limitations of this approach. Ehab introduced an approach based on BDDs (Binary Decision Diagrams) and contended that BDDs are a more effective way to formalize the rules; he showed how conjunctions and disjunctions of rules are expressed using BDDs. He also showed how IPSec policies can be represented using BDDs and how one can verify policy conflicts for paths and compositions. Ehab presented how to use BDDs to model routing configurations. The developed tool is available for download from Ehab's Web site.

Question (Mark Burgess): Have you considered using ontology or description logic?

Answer: We tried a little bit, but BDD is a mature area and we can leverage on its maturity.

Question: How useful would a description logic be instead of BDD for this work?

Answer: It could be more expressive, but the vast literature and research already done on BDD means that we can leverage on the success of BDD as a proven tool.

- ***WISE: Predicting Service Response Times in "What-If" Deployment Scenarios***

  Mukarram Bin Tariq, Georgia Tech

Mukarram presented a tool for evaluating the effect of response time distribution for hypothetical deployment scenarios for content distribution networks. The deployment scenarios include deployment of new data centers, changing DNS mapping for subsets of clients, and having new peering with a new ISP. For networks such as content distribution networks, no accurate model for response time exists, and it is generally hard to develop such models, owing to the scale and complexity of the system. Mukarram showed how machine learning can be used to model response time as a function of variables that can be easily observed in existing deployments; further, he presented how interactions among the observed variables can be captured in the form of a causal Bayesian network and be subsequently used to evaluate "what-if" scenarios. The What-If Scenario Evaluator (WISE) includes a high-level, declarative specification language called WISE-SL, which the network designers can use to express in a very succinct manner the scenarios they wish to evaluate. Mukarram also presented results on accuracy and effectiveness of WISE predictions based on dataset and events observed in Google's Web search service delivery network.

Question: How is the WISE approach better than doing, say, NS simulations?

Answer: Generally, it is difficult to make accurate simulations for the kinds of large, complex systems that we are talking about here. Trace-driven simulations can help somewhat in terms of input to the system, but still we need to model the system accurately, which is hard. The WISE-based approach is good in the sense that it does not require explicit modeling of the system.

Question: To what network scenarios can WISE be applied?

Answer: The cases where there are no hidden variables that can affect a scenario can be easily evaluated with this approach.

- ***MulVAL: A Logic-Based, Data-Driven Enterprise Security Analyzer***

  Xinming (Simon) Ou, Kansas State University

MulVAL presents a logic-based approach for security analysis in multihost networks. The formal definitions of security risks from OVAL and the National Vulnerability Database (NVD) serve as input (base data) to the MulVAL tool. The interactions within a network are formulated as rules in Datalog. Rules are completely independent of the network setting and are generic enough to be applied to any network. The tool can detect multistage attacks in multihost networks. The cause of an attack is represented through an attack graph.

Question: How is analysis done over multiple stages in the network?

Answer: The interaction rules are written in Datalog and multistage attacks are formulated in Datalog using recursion.

Question: How fast is MulVAL analysis?

Answer: All the interaction rules for Linux are written using approximately 20 Datalog rules. The time it takes to perform the analysis is quadratic in terms of the machines in the network. For large networks the analysis takes a few seconds and generation of attack graphs takes about 1–2 hours.

- ***Maestro: A New Architecture for Realizing and Managing Network Controls***

  T.S. Eugene Ng, Rice University

There is a lack of interface and abstraction for coordination among protocols. Eugene proposed an OS that over-sees the whole network. The idea is to insert safety-checking applications in the network to look for network misbehaviors, such as routing loops and security breaches. The proposed OS for a network runs the logic of routing, and routers only perform forwarding. Further, a "BIOS" is required for the network. Maestro is an operating platform that provides an API to the applications and an interface for network state exchange (BIOS). Routers essentially work as "sensors" that measure the state of the network; the state is presented as Network Views and gets passed to the applications. Applications are stateless functions that can form an Application DAG (a sequence of applications). The Application DAG is triggered by some prespecified triggers.

Question: How does Maestro contrast to InfiniBand?

Answer: We have not specifically contrasted with their approach.

Question (Burgess): Is it tailored somehow to BGP?

Answer: An OS controls one administrative network.

Question: What about security implications as well as delegations?

Answer: We are looking into that.

■ ***Request and Response Monitoring of Online Services***

Juhan Lee, Microsoft

The goal of this project is to monitor request and response streams for massive online services. Typical monitoring systems do not scale to the requirements of large, complex systems such as MSN online services, where there are thousands of servers in the network.

In the presented scheme, an application overlay network is used for request-response exchange. A token-based approach is used for scheduling requests. A token is generated when a new request arrives. The generated token is passed along the servers serving the request and application-specific logs are generated. The logs of the requests and their responses are enormous. It is difficult to store such large logs. Conditional logging is used to reduce the storage requirements. Correlated sampling is used to lower the sampling rate for request-response. Lee showed a couple of examples of scenarios where their technique was able to use the logs to point out problems in the services, in particular a case where information was being served in an incorrect language at an international portal of MSN services.

Question: Is correlated sampling done among domains or across the entire network?

Answer: Sampling is done across the entire network and all the components. It allows us to pinpoint what's taking the most time. In the MSN publishing platform (portal), multiple requests (asynchronous requests) are sampled in a single session. The MSN publishing platform is incrementally deployable.

Question: What configuration errors are detected?

Answer: Generally, misconfigurations that can lead to unexpected application behavior can be detected.

■ ***Panel on Security Configuration***

Moderator: Steve Bellovin, Columbia University; attendees as panelists

Configuration is important for firewalls, depends on what services are being run and on whether the service set is contingent on the versions or patch levels, and has implications for authorized parties changing configuration and how one manages the authorization list.

There are various security scenarios to be considered. (1) Appliance (firewalls, filtering routers, etc): What should the configuration be in a complex topology? Typical corporate networks have several entry points into the networks. How do we reconcile different policy needs? (2) Infrastructure: Do the infrastructure nodes have proper security configuration? How do you know whether some element's configuration is wrong? (3) Servers: What services are they running? What versions? How do you monitor changes, new nodes, etc.? (4) Personal machines: How do you balance personal needs versus corporate needs? How do you enforce or prevent upgrades? How do you change the configuration of a laptop or home computer? How do you balance the need for central control with what cannot be enforced?

Mark Burgess asked whether control is the same as security, especially if it is not enforceable. Steve and other participants noted that exerting control is a way to enforce security; it is necessary but not sufficient. Andrew Hume equated it with ensuring border security and immigration issues. Mark remarked that we should think of security in civil society, where if most of us agree to abide by a law, then it makes it easier to enforce it.

Steve directed the participants to focus back on configuration issues of security. Assuming a certain policy of controlling, how do you enforce it? One of the participants remarked that there are various constraints, which may not be overspecified, and certain things are left free. How do you compose such constraints? Steve asked whether there is sufficient homogeneity in configuration and devices to allow such compatibility.

The ensuing discussion again drifted toward the inconvenience that security poses and causes people to "disable security." The problem arises because of mismatch of values between the system administrators and the users of the network. One of the participants remarked that usually there is a gap between what a human administrator wishes to achieve and what gets translated into configuration.

Ehab observed that a necessary component that is usually not explicitly evaluated in configuration management is risk assessment; if configuration management is integrated with such assessment, it will lead to consistent and sensible configurations.

Steve Bellovin asked the participants to consider the question of how to introduce new technologies and how to deal with new applications in terms of configuration. Paul Anderson affirmed the need to state goals and criteria, for the criteria, not the specific images of OSes, are what we wish to enforce. And we do not want to specify everything else. Ehab remarked that, generally speaking, creating a program from a specification is not solvable; are we going that route? Andrew commented that because of this problem, we are stuck with configurations that work, or a certain subset of use cases that have been tested.

Mark believes that it should not be so hard to manage the variances if we are not afraid of sophistication, specifically, dealing in probabilities. We tend to operate in paranoid mode, where we want to address issues that are highly improbable, which leads to complex security configurations that are difficult to work with. Ehab also asked whether there have been any incidents of remote malicious configuration change.

Steve Bellovin shepherded the audience back to the topic, refocusing on two issues: (1) keeping track of only the authorized users and what changes they make (i.e., managing the list of users and what they are allowed to do); (2) what to do when someone breaks in through a hack.

Sanjai added that composability is important for abstraction. A declarative approach and the inference engine will allow us to verify whether two rules are in conflict. Mark mentioned IETF BDIM WG activity, which is looking at how to convert high-level goals into low-level policy.

■ ***Automata-Driven Techniques for Managing Firewall Configuration***

Alok Tongaonkar, Stony Brook University

Alok talked about optimizing the performance of firewall-rules analysis using automata-based techniques. He talked about syntax- and semantics-driven analysis techniques for firewall configuration analysis. Rule interaction makes the analysis difficult. Alok mentioned that the BDD-based techniques proposed by Ehab (the first session) are semantics-driven.

Alok told how a finite state automata (FSA) is built for packet classification. The enormous state space of a packet is divided into finite regions of the automata. Packet space is divided into regions based on their match with the firewall rules. Priorities of rules can govern the classification (i.e., application to a packet). He also discussed shadowing of rules. The rules are analyzed, and intersections and shadowing among them are found. FSA size explosion is possible because of duplication of rules that may occur from ranges and "less than" and "greater than" occurrences in the rules. Their algorithm minimizes the duplication of rules. The FSA is built incrementally using candidate (probable) and match (matched) sets, resulting in a compact automata.

Sanjai: Why do this kind of analysis? Why should we not just build right configurations to begin with?

Answer: "Evolution" of rules may inadvertently result in conflicts and misconfiguration.

Steve: Should we have a better abstraction than priority-based mechanisms?

Answer: We are trying to convert priority-based rules into nonpriority-based rules; however, this results in an explosion of the number of rules.

Steve: Is a human factor involved here and does the explosion make it more or less comprehensible?

Answer: It is not clear at this stage.

Sanjai: Are these rule lookups $O(n)$ and are there really that many rules?

Answer: Yes; for priority-based rules it has to be.

Steve affirmed that there are indeed many rules for security configuration and referenced a study by Arbor Networks.

■ ***Vulnerability Analysis via Deductive Spreadsheets (DSS)***

Anu Singh, Stony Brook University

Anu explained the desired properties of a security policy analysis tool. She explained the Role Based Access Control (RBAC) model as background. The current prototype implementation of DSS, called XcelLog, is built as an add-in to MS Excel. The formula language of DSS supports sets and tuples. Recursive relations can be represented by using DSS. XcelLog uses XSB (Prolog) as the underlying evaluation engine. The features of DSS include highlighting of explanations for results and incremental evaluation. Anu gave a demo of the DSS using the RBAC example. She also showed how to do vulnerability analysis for a multihost network in DSS.

Question: If there is more than one way of getting to a condition (attack), will DSS be able to highlight?

Answer: The tool can find multiple ways of getting to a condition (attack), but it may not be able to distinguish among them.

Question: What are the relations between DSS cells and prolog predicates?

Answer: DSS expressions represent logical conditions using cell references. Anu explained with a demo.

■ ***An Analysis of the VLAN Design of an Operational Campus Network***

Yu-Wei Sung, Purdue University

Yu-Wei talked about generation of task-driven network-wide abstractions to configure enterprise network design. He emphasized the need for abstraction of a network design by elevating the observations to abstraction that would simplify the design. It is important to understand the intent of the operator, he added. VLAN configuration is error-prone and time-consuming, which is why abstraction of the VLAN is useful. VLANs can be abstracted by logical grouping of hosts. The key components in a VLAN configuration are the access port and trunk ports not directly connected to hosts serving as carriers for other VLANs. Abstractions model the network as a topology of hosts and switches, and a router placement strategy determines where to place the routers.

Question: How does the firewall interact with VLAN?

Answer: This hasn't been considered yet.

Question: What data size is used?

Answer: A single network with 1300 switches.

Question: Can the tool generate configuration consistent with the abstract model?

Answer: It can generate useful recommendations that can help the operator in network design.

■ *Fixing Configuration Errors via Unsatisfiability Analysis*

Sanjai Narain, Telcordia Technologies, Inc.; Daniel Jackson, MIT; Sharad Malik, Princeton University

Security cannot be divorced from functionality. Since there are usually so many interweaving requirements, we can put all the various requirements in a melting pot and generate a configuration from that. The specific problem addressed in the talk was fixing configuration errors via unsatisfiability analysis. Sanjai explained, through an example, how security and reliability are interrelated, where if a separate IPSec tunnel is not established for the backup router, the communication would break down even if the backup router took over.

Sanjai discussed how to specify security, routing, and reliability in a single unified framework and how to do it efficiently. He presented a requirement solver that takes as input a specification in first-order logic and the configuration variables database and produces the configurations. ALLOY, a first-order language, is used for specifying the constraints, which are solved using SAT solver. Once the requirements are, at a high level, expressed as FOL constraints, the Un-SAT-core (unsatisfiability) finds the subset of the sets that are unsatisfiable. A counterexample can be obtained from the result of the constraint solver. The next step is to find the configuration variables that violate the constraint and then relax the constraints and re-solve it.

The issue of scalability arises if the constraints are specified in an obvious way. The aim is to scale this. The requirements are preprocessed and constraints are partially evaluated using the constraint SAT solver. This makes it practical to apply to large network configurations. The system uses a deductive spreadsheet to specify constraints on cell values and display results.

Question: How helpful is this analysis for network configuration management?

Answer: Host-level configurations can be modeled and analyzed using this framework.

Question: What about constraints crossing the layers of the protocol stack (application layer versus the network layer)?

Answer: The solver can capture all dependencies, including cross-layer dependencies.

Question: Does the solver give the best answer or just an answer?

Answer: Presently it gives an answer. If the notion of "best" can be formalized as a constraint, then it can give the best answer.

■ *Panel on Autonomic Configuration*

Moderator: John Strassner, CTO Motorola; attendees as panelists

John Strassner gave a brief introduction to autonomic networking, which he described as the process of offloading simpler automatable things to automated processes. He contended that operators are losing money because the OSS (Operations Support System) is too complicated to address business and customer needs. He said that we want to build something that takes care of autonomic functions, so that the human in charge has less to do. We want the system to "learn" how to do everyday things. He gave an overview of the system being built by his group. The system uses machine-learning techniques. [Editor's note: Strassner's talk was similar to his keynote.]

Andrew: Autonomics are applicable to simple and very well-defined tasks, such as breathing or heart pumping.

John: If there is a set of transformations that can take the service and business goals, along with the environments, and give out CLI commands and configuration, then this is autonomics. Our research is about merging ontology with CLI-level stuff. It is like a "multigraph." At the higher level of the graph there is a different interpretation of "errors" than at the lower layer; the following steps are involved in such a system: IOS → Model-based translation → ACML → Analyze data and event and determine actual state → If not desirable → Reconfigure → Repeat.

Andrew: Telephony built a similar network to map errors to the customer, but this mapping has to be dynamic because the relationships are fluid. We live at a 99.99% uptime level, but that does not have too much bearing on how the operator is doing at a business level.

Sanjai Narain: The emphasis in this work seems to be on performance; aren't things like security configuration just as important? And in that case, how do you see the system learning its state and changing the configuration?

John: Learning that kind of semantics and subsequent configurations can be hard but, given higher-level goals, it can be achieved.

■ *Advanced Topics Workshop*

Summarized by Josh Simon (jss@clock.org)

The Advanced Topics Workshop was once again hosted, moderated, and refereed by Adam Moskowitz. We started with cable management 101 in separating the large bundle of CAT-5 cable into strands for us to connect our laptops to the local network switch, as there are enough of us that we overload the wireless access point. We followed that with introductions around the room. For a variety of reasons, several of the Usual Suspects weren't at this year's workshop. Despite this, in representation, businesses (including consultants) outnumbered universities by about four to one; over the course of the day, the room included six LISA program chairs (four past, present, and future; up from three last year) and 11 past or present members of the USENIX, SAGE, or LOPSA Boards (up from five last year).

Setting up involved untangling the bundled CAT-5 cables, connecting them to attendees' laptops and the local switch, getting the moderation software up and running, setting the correct time zone on the server, and so on.

Our first topic was on management versus technology. About half of the room were either full- or part-time managers, and we discussed some of the problems we have interacting with our management. Some of the concerns were knee-jerk, too-shiny managers, cultural differences when your manager is in another country, and managers who used to be in sales positions. Some folks discussed their specific situations and asked for advice in solving them. One common suggestion was to communicate differently; remember that managers (especially those on the financial side who approve capital budgets) tend to speak business-speak and not "techie." They don't care about the new gee-whiz neato-peachy-keen technology but, rather, in how this new thing will solve their problems and provide a decent return on investment.

A side discussion took place on cultural issues that differ from the North American standard most of us are used to, and how that can affect communication styles as well as resumes.

After the morning break, we discussed career concerns. Most of the people in the room had 15 or more years of experience, and many of us had more than 20 years of experience. Assuming that retirement isn't an option (for whatever reason, be it financial or boredom), what's the right thing to do if you wind up looking for work? One person discussed how he neglected to ask questions of the company during the interview process; after accepting the offer and working for some length of time, he realized he was a bad fit for the position. One suggestion for avoiding this in the future was to ask better questions before accepting any offer; another suggestion was to consider a contract-to-permanent position, since it gives both parties an out without the company having to let a senior person go. One topic that fell out of this is whether there's a technical growth path at your company or whether "senior" implies a management position. Another topic was the technology refresh rate for individuals and whether staying generalists or becoming specialists was the better course of action. (Consensus seemed to be for the former.) Those who are retiring in the fairly near future have to make peace with what's good enough and remember that "good enough" isn't necessarily the same as settling. Do what needs to be done and find enjoyment in that. Whatever you're doing, remember to work, to play, and to live, not just to exist. Do things to keep yourself interested and awake at your job; don't just settle into a rut.

We next discussed patterns as an abstraction layer in system administration. There's apparently some controversy in patterns; some think they're a good way to abstract problems and provide a common shorthand; others think they're not worth the electrons and doubt they're applicable to system administration.

After our lunch break, we discussed things we should have done differently. One example was the whole IPv6 rollout. A lot of places don't see any need to deploy it and wonder whether ignoring it now will cause problems later. CFOs don't see the benefit from or ROI in another technology refresh. Widespread adoption of something new requires there be some kind of benefit on the business level, and right now businesses don't tend to see a need for IPv6. Right now, there is very little IPv6-only content out there; if services or content were made IPv6-only, that could drive folks to convert, assuming that their equipment is IPv6-capable (e.g., not all SOHO equipment is).

We next had a brief discussion on the different uses of DNS and search engines. Both are treated as a way to find resources: DNS is a way of finding IP address-to-name mappings and search engines are a way of finding some specific document or site.

We next went around the room to discuss our favorite new-to-us tools from the past year. Common examples were AFS and ZFS, certain KVM cards, load balancers, ntop, Puppet, Ruby and jRuby, spam management tools, svk, tcptrace, the iPhone, Time Machine in Mac OS 10.5, virtualization, and zones in Solaris. Several people chose Puppet for their configuration management, mainly because it was faster to get it up and running and it had a sufficiently low learning curve for installation and configuration.

Our next discussion was on virtualization. At least one participant has done nothing with it and wondered if it was worthwhile; the consensus seemed to be that there are some areas where it's not a benefit, such as in a high-performance computing environment. Someone plugged this year's refereed paper, "Decision Support for Virtual Machine Re-Provisioning in Production Environments," by Kyrre Begnum, Matthew Disney, Æleen Frisch, and Ingard Mevåg, for performance statistics. Some are only using virtualization in nonproduction environments.

We next talked about delegating identity management. The example environment is a department within a university that uses automated identity management to manage authentication and authorization controls for students, faculty, staff, alumni, and guests (e.g., investigators on research grants from other universities). The central IT organization can provide some of the information they need, but not all of it. The question of how they can cascade information management systems was addressed. The short answer is that processes need to be put into place to incorporate the data to allow for both provisioning and revocation and to make sure essential safeguards are in place such that a failure upstream (e.g., HR's database failing miserably) doesn't accidentally cause a disaster downstream (e.g., the deletion of all staff accounts).

The next major discussion concerned distributed or geographically disparate personnel. We talked about what server infrastructure needed to be used in remote data centers for part-time workers; the answer generally depends on how many people and how much network traffic there is. Items to consider are VOIP or POTS phones, home directory access over the WAN, docking stations, printers, reserved offices or cubes ("hoteling"), and possibly a conference room, depending on the size of the office and the data center. We also talked about tools for communication and collaboration; many use some form of instant messenger client (such as internal IRC channels or logged IM conversations), email, trouble ticketing systems, and wikis. If you are using any of these technologies, remember to include them in (as a critical part of, where need be) your disaster recovery planning.

Our final discussion was a quick around-the-room on the cool tool for next year. Ruby, Solaris 10, and virtualization were the most commonly mentioned, with configuration management tools, Perl 6, VOIP, wiki deployment, and ZFS rounding out the list.

Announcement and Call for Participation

22nd Large Installation System Administration Conference (LISA '08)

Sponsored by USENIX and SAGE

http://www.usenix.org/lisa08

November 9–14, 2008 **San Diego, CA, USA**

Important Dates

Extended abstract and paper submissions due: *May 8, 2008, 11:59 p.m. PDT*
Invited talk and workshop proposals due: *May 20, 2008*
Guru Is In and Hit the Ground Running proposals due: *May 31, 2008*
Notification to authors: *Mid-June 2008*
Poster proposals due, first round: *July 16, 2008*
Notification to poster presenters, first round: *July 23, 2008*
Final papers due: *August 20, 2008*
Poster proposals due, second round: *October 22, 2008*
Notification to poster presenters, second round: *October 29, 2008*

Conference Organizers

Program Chair
Mario Obejas, *Raytheon*

Program Committee
Paul Anderson, *University of Edinburgh*
Derek Balling, *Answers Corporation*
Travis Campbell, *AMD*
Narayan Desai, *Argonne National Laboratory*
Æleen Frisch, *Exponential Consulting*
Peter Baer Galvin, *Corporate Technologies*
Brent Hoon Kang, *University of North Carolina at Charlotte*
Chris McEniry, *Sony Computer Entertainment America*
David Parter, *University of Wisconsin*
David Plonka, *University of Wisconsin*
Melanie Rieback, *Vrije Universiteit*
Kent Skaar, *Bladelogix*
Chad Verbowski, *Microsoft*

Invited Talks Coordinators
Rudi van Drunen, *Competa IT/Xlexit*
Philip Kizer, *Estacado Systems*

Workshops Coordinator
Lee Damon, *University of Washington*

Guru Is In Coordinator
John "Rowan" Littell, *California College of the Arts*

Hit the Ground Running Coordinator
Adam Moskowitz, *Permabit Technology Corporation*

Work-in-Progress Reports and Posters Coordinators
Brent Hoon Kang, *University of North Carolina at Charlotte*
Gautam Singaraju, *University of North Carolina at Charlotte*

Overview

Since 1987, the annual LISA conference has become the premier meeting place for professional system and network administrators. System administrators of all ranks, from novice to veteran, and of all specialties meet to exchange ideas, sharpen skills, learn new techniques, debate current issues, and mingle with colleagues and friends.

Attendees are diverse, a rich mix of nationalities and of educational, government, and industry backgrounds. We work in the full spectrum of computing environments (e.g., large corporations, small businesses, academic institutions, government agencies). We include full- and part-time students engaged in internships, as well as students and faculty deeply involved in system administration research. Whereas many attendees focus on practical system administration, others focus on speculative system administration research. We support a broad range of operating systems (e.g., Solaris, Windows, Mac OS X, HP-UX, AIX, BSD, Linux) and commercial and open source applications, and we run them on a variety of infrastructures.

The conference's diverse group of participants are matched by a broad spectrum of conference activities:

- A training program for both beginners and experienced attendees covers many administrative topics, ranging from basic procedures to using cutting-edge technologies.
- Refereed papers present the latest developments and ideas related to system and network administration.
- Workshops, invited talks, and panels discuss important and timely topics in depth and typically include lively and/or controversial debates and audience interaction.
- Work-in-Progress Reports (WiPs) and poster sessions provide brief looks ahead to next year's innovations.
- The Hit the Ground Running track presents multiple important topics in single sessions, distilled down to a few solid points.

LISA also makes it easy for people to interact in more informal settings:

- Noted experts answer questions at Guru Is In sessions.
- Participants discuss/celebrate/commiserate about a shared interest at Birds-of-a-Feather (BoF) sessions.
- Vendors answer questions and offer solutions at the Exhibition.

Finally, we strongly encourage informal discussions among participants on both technical and nontechnical topics in the famous "hallway track." LISA is a place to learn and to have fun!

Get Involved!

The theme for LISA '08 is "Real World System Administration."

Experts and old-timers don't have all the good ideas. We welcome participants who will provide concrete ideas to immediately implement, as well as those whose research will forge tomorrow's computing infrastructures. We are particularly keen to showcase novel solutions or new applications of mature technologies. This is your conference, and we want you to participate. Here are examples of ways to get involved in this 22nd LISA conference:

- Submit a draft paper or extended abstract for a refereed paper.
- Suggest an invited talk or panel discussion.
- Propose a short Hit the Ground Running presentation.
- Share your experience by leading a Guru Is In session.
- Create and lead a workshop.
- Propose a tutorial topic.
- Present a Work-in-Progress Report (WiP) or submit a poster.
- Organize a Birds-of-a-Feather (BoF) session.
- Email an idea to the Program Chair: lisa08ideas@usenix.org.

Refereed Papers

Effective administration of a large site requires a good understanding of modern tools and techniques, together with their underlying principles—but the human factors involved in managing and applying these technologies in a production environment are equally important. Bringing together theory and practice is an important goal of the LISA conference, and practicing system administrators, as well as academic researchers, all have valuable contributions to make. A selection of possible topics for refereed papers appears in a separate section below, but submissions are welcome on any aspect of system administration, from the underlying theory of a new configuration technique to a case study on the management of a successful site merger.

Whatever the topic, it is most important that papers present results in the context of current practice and previous work: they should provide references to related work and make specific comparisons where appropriate. The crucial component is that your paper present something new or timely; for instance, something that was not previously available, or something that had not previously been published. Careful searching for publications on a similar theme will help to identify any possible duplication and provide pointers to related work; the USENIX site contains most previous LISA conference proceedings, which may provide a starting point when searching for related publications: http://www.usenix.org/events/byname/lisa.html.

Cash prizes will be awarded at the conference for the best refereed paper as well as for the best refereed paper for which a student is the lead author; a special announcement will also be made about these two papers.

Proposal and Submission Details

Anyone who would like help in writing a proposal should contact the program chair at lisa08chair@usenix.org. The conference organizers are keen to make sure that good work gets published, and we are happy to help at any stage in the process.

Proposals may be submitted as draft papers or extended abstracts.

- **Draft papers:** This is the preferred format. A draft paper proposal is limited to 16 pages, including diagrams, figures, references, and appendices. It should be a complete or near-complete paper, so that the Program Committee has the best possible understanding of your ideas and presentation.
- **Extended abstracts:** An extended abstract proposal should be about 5 pages long (at least 500 words, not counting figures and references) and should include a brief outline of the final paper. The form of the full paper must be clear from your abstract. The Program Committee will be attempting to judge the quality of the final paper from your abstract. This is harder to do with extended abstracts than with the preferred form, draft papers, so your abstract must be as helpful as possible in this process to be considered for acceptance.

Paper authors are also invited to submit posters, as outlined below, to accompany their presentations; these provide an overview of the work and a focal point for delegates to meet with the author.

General submission rules:

- All submissions must be electronic, in ASCII or PDF format only. Proposals must be submitted using the Web form located on the LISA '08 Call for Papers Web site, http://www.usenix.org/lisa08/cfp.
- Submissions whose main purpose is to promote a commercial product or service will not be accepted.
- Submissions may be submitted only by the author of the paper. No third-party submissions will be accepted.
- All accepted papers must be presented at the LISA conference by at least one author. One author per paper will receive a registration discount of $200. USENIX will offer a complimentary registration for the technical program upon request.
- Authors of an accepted paper must provide a final paper for publication in the conference proceedings. Final papers are limited to 16 pages, including diagrams, figures, references, and appendices. Complete instructions will be sent to the authors of accepted papers. To aid authors in creating a paper suitable for LISA's audience, authors of accepted proposals will be assigned one or more shepherds to help with the process of completing the paper. The shepherds will read one or more intermediate drafts and provide comments before the authors complete the final draft.
- Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, to ensure the integrity of papers under consideration, program committees may share information about submitted papers with other conference chairs and journal editors. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case. Authors uncertain whether their submission meets USENIX's guidelines should contact the program chair, lisa08chair @usenix.org, or the USENIX office, submissionspolicy@usenix.org.
- Papers accompanied by nondisclosure agreement forms will not be considered. All submissions will be treated as confidential prior to publication in the Proceedings.

For administrative reasons, every submission must list:

1. Paper title, and names, affiliations, and email addresses of all authors. Indicate each author who is a full-time student.
2. The author who will be the contact for the Program Committee. Include his/her name, affiliation, paper mail address, daytime and evening phone numbers, email address, and fax number (as applicable).

For more information, please consult the detailed author guidelines at http://www.usenix.org/events/lisa08/cfp/guidelines.html. **Paper and extended abstract submissions are due by 11:59 p.m. PDT on May 8, 2008.** Authors will be notified by mid-June whether their papers have been accepted.

Training Program

LISA offers state-of-the-art tutorials from top experts in their fields. Topics cover every level from introductory to highly advanced. You can choose from over 50 full- and half-day tutorials ranging from Linux-HA, through performance tuning, Solaris, Windows, Perl, Samba, network troubleshooting, security, network services, filesystems, backups, Sendmail, spam, and legal issues, to professional development.

To provide the best possible tutorial offerings, USENIX continually solicits proposals and ideas for new tutorials, especially on subjects not yet covered. If you are interested in presenting a tutorial or have an idea for a tutorial you would like to see offered, please contact the Education Director, Daniel V. Klein, at tutorials@usenix.org.

Invited Talks

An invited talk discusses a topic of general interest to attendees. Unlike a refereed paper, this topic need not be new or unique but should be timely and relevant or perhaps entertaining. A list of suggested topics is available in a separate section below. An ideal invited talk is approachable and possibly controversial. The material should be understandable by beginners, but the conclusions may be disagreed with by experts. Invited talks should be 60–70 minutes long, and speakers should plan to take 20–30 minutes of questions from the audience.

Invited talk proposals should be accompanied by an abstract of less than one page in length describing the content of the talk. You can also propose a panel discussion topic. It is most helpful to us if you suggest potential panelists. Proposals of a business development or marketing nature are not appropriate. Speakers must submit their own proposals; third-party submissions, even if authorized, will be rejected.

Please email your proposal to lisa08it@usenix.org. **Invited talk proposals are due May 20, 2008.**

The Guru Is In

Everyone is invited to bring perplexing technical questions to the experts at LISA's unique Guru Is In sessions. These informal gatherings are organized around a single technical area or topic. Email suggestions for Guru Is In sessions or your offer to be a Guru to lisa08guru@usenix.org. **Guru Is In proposals are due May 31, 2008.**

Hit the Ground Running

This track consists of five high-speed presentations packed into each 90-minute session. The presentations are intended to give attendees a "brain dump" on a new technology, new features in an existing protocol or service, an overview of the state of the art of a technique or practice, or an introduction to an existing technology that is becoming more widely used.

HTGR proposals should be accompanied by an abstract of less than one page in length describing the content of the talk. Proposals of a business development or marketing nature are not appropriate. Speakers must submit their own proposals; third-party submissions, even if authorized, will be rejected. Suggestions for desired HTGR presentations are also welcome (if possible, accompanied by a suggestion for a speaker).

Please email your proposal to lisa08htg@usenix.org. **HTGR proposals are due May 31, 2008.**

Workshops

One-day workshops are hands-on, participatory, interactive sessions where small groups of system administrators have an opportunity to discuss a topic of common interest. Workshops are not intended as tutorials, and participants normally have significant experience in the appropriate area, enabling discussions at a peer level. However, attendees with less experience often find workshops useful and are encouraged to discuss attendance with the workshop organizer.

A workshop proposal should include the following information:

- Title
- Objective
- Organizer name(s) and contact information
- Potential attendee profile
- Outline of potential topics

Please email your proposal to lisa08workshops@usenix.org. **Workshop proposals are due May 20, 2008.**

Posters

This year's conference will include a poster session. This is an opportunity to display a poster describing recent work. The posters will be on display during the conference, and fixed times will be advertised when authors should be present to discuss their work with anyone who is interested. This provides a very good opportunity to make contact with other people who may be interested in the same area. Student posters, practitioners sharing their experiences, and submissions from open source communities are particularly welcome.

To submit a poster, please send a 1–5 page proposal or 6–12 PowerPoint slides in PDF to lisa08posters@usenix.org. Please include your name, your affiliation, and the title of your poster. There will be two rounds of submission and review of poster proposals. You may submit your poster during either the first or the second round.

The first deadline for submissions is July 16, 2008. Please submit your poster by this deadline if you plan to apply for a student conference grant or will be traveling to LISA '08 from outside the United States and need to allow time for visa preparation. Accepted poster authors from the first round will be notified by July 23.

The second deadline for submissions is October 22. Accepted poster authors from the second round will be notified by October 29. Completed posters from both rounds will be required by the start of the conference.

Poster presenters who would also like to give a short presentation may also register for a WiP as below.

Work-in-Progress Reports (WiPs)

A Work-in-Progress Report (WiP) is a very short presentation about current work. It is a great way to poll the LISA audience for feedback and interest. We are particularly interested in presentations of student work. To schedule a short presentation, send email to lisa08wips@usenix.org or sign up on the first day of the technical sessions.

Birds-of-a-Feather Sessions (BoFs)

Birds-of-a-Feather sessions (BoFs) are informal gatherings organized by attendees interested in a particular topic. BoFs will be held in the evening. BoFs may be scheduled in advance by emailing bofs@usenix.org. BoFs may also be scheduled at the conference.

Possible Topics for Authors and Speakers

Technical Challenges

- Authentication and authorization: "Single sign-on" technologies, identity management
- Autonomic computing: Self-repairing systems, zero administration systems, fail-safe design
- Configuration management: Specification languages, configuration deployment
- Data center design: Modern methods, upgrading old centers
- Data management: DBMS management systems, deployment architectures and methods, real world performance
- Email: Mail infrastructures, spam prevention
- Grid computing: Management of grid fabrics and infrastructure
- Hardware: Multicore processor ramifications
- Mobile computing: Supporting and managing laptops and remote communications
- Multiple platforms: Integrating and supporting multiple platforms (e.g., Linux, Windows, Macintosh)
- Networking: New technologies, network management
- Security: Malware and virus prevention, security technologies and procedures, response to cyber attacks targeting individuals
- Service
- Standards: Enabling interoperability of local and remote services and applications
- Storage: New storage technologies, remote filesystems, backups, scaling
- Web 2.0 technologies: Using, supporting, and managing wikis, blogs, and other Web 2.0 applications
- Virtualization: Managing and configuring virtualized resources

Professional Challenges

- Budgeting: Definitions and methods
- Communication: Tools and procedures for improving communication between administrators and users, distribution organizations, or teams
- Consolidation: Merging and standardizing infrastructures and procedures
- Devolution: Managing dependence on devolved services (calendars, mail, Web 2.0, etc.) and users
- Ethics: Common dilemmas and outcomes
- Flexibility: Responding effectively to changes in technology and business demands
- In-house development: The (dis)advantages and pitfalls of in-house technology development
- Legislation: Security, privacy
- Management: The interface and transition between "technical" and "managerial"
- Metrics: Measuring and analyzing the effectiveness of technologies and procedures
- Outsourcing/offshoring system administration: Is it possible?
- Proactive administration: Transitioning from a reactive culture
- Standardizing methodologies: Sharing best practice
- Training and staff development: Developing and retaining good system administrators; certifications
- User support: Systems and procedures for supporting users

Contact the Chair

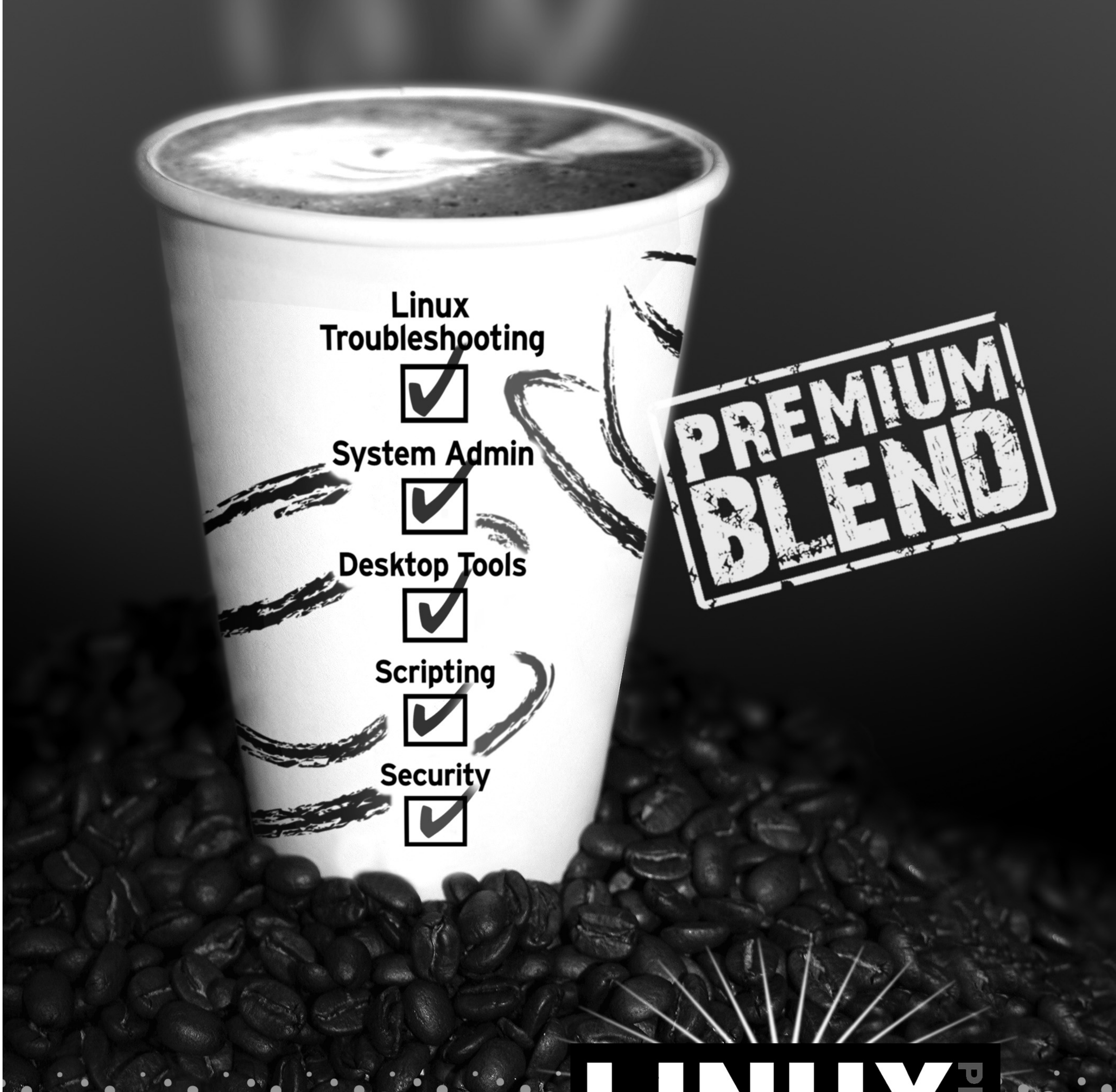
The program chair, Mario Obejas, is always open to new ideas that might improve the conference. Please email your ideas to lisa08ideas@usenix.org.

Final Program and Registration Information

Complete program and registration information will be available in August 2008 at the conference Web site, http://www.usenix.org/lisa08. If you would like to receive the latest USENIX conference information, please join our mailing list at http://www.usenix.org/about/mailing.html.

Sponsorship and Exhibit Opportunities

The oldest and largest conference exclusively for system administrators presents an unparalleled marketing and sales opportunity for sponsoring and exhibiting organizations. Your company will gain both mind share and market share as you present your products and services to a prequalified audience that heavily influences the purchasing decisions of your targeted prospects. For more details please contact exhibits@usenix.org.


Linux Troubleshooting
System Admin
Desktop Tools
Scripting
Security
PREMIUM BLEND
LINUX PRO MAGAZINE

SD WEST 2008

Software Development Conference & Expo West

March 3-7, 2008

Santa Clara Convention Center
Santa Clara, CA

SUPER EARLY BIRD DISCOUNT
Register by JAN 11
SAVE UP TO $500

EARLY BIRD DISCOUNT
Register by FEB 8
SAVE UP TO $300

JOIN THE REVOLUTION

FEATURING OVER 200 EDUCATIONAL SESSIONS

PLUS:
Keynotes, Expo, Birds-of-a-Feathers, Panels, Case Studies, Parties and other Special Events!

EDUCATIONAL SESSIONS INCLUDE:

LANGUAGES & PLATFORMS:
- C++
- JAVA PROGRAMMING
- .NET DEVELOPMENT
- RUBY

NEXT-GEN WEB:
- WEB 2.0/WEB DEV
- WEB SERVICES/SOA

SOFTWARE DEVELOPMENT LIFECYCLE:
- MODELING & DESIGN
- PEOPLE, PROCESS & METHODS
- REQUIREMENTS & ANALYSIS
- SECURITY
- TESTING & QUALITY

SOFTWARE BUSINESS & CAREER:
- BUSINESS OF SOFTWARE
- NEW IT AND THE DEVELOPER
- NEW EMERGING TECHNOLOGY

REGISTER TODAY AT WWW.SDEXPO.COM

CMP
United Business Media

nsdi '08

http://www.usenix.org/nsdi08

April 16–18, 2008, San Francisco, CA

Sponsored by USENIX in cooperation with ACM SIGCOMM & ACM SIGOPS

The 5th USENIX Symposium on Networked Systems Design and Implementation will focus on the design principles of large-scale networks and distributed systems. Join researchers from across the networking and systems community in fostering cross-disciplinary approaches and addressing shared research challenges.

Don't miss these co-located workshops:

- Usability, Psychology, and Security 2008 (UPSEC '08), April 14, 2008
- First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '08), April 15, 2008
- Workshop on Organizing Workshops, Conferences, and Symposia for Computer Systems (WOWCS '08), April 15, 2008

http://www.usenix.org/nsdi08/workshops

USENIX
The Advanced Computing
Sytems Association

Early Bird Registration Deadline: Monday, March 24, 2008

;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