

;login:

THE MAGAZINE OF USENIX & SAGE

December 2000 • volume 25 • number 8

inside:

OVERVIEW

Needles in the Craystack: When
Machines Get Sick

National Laws and the Net

Musings

PROGRAMMING

The Tclsh Spot
and more . . .

SECURITY

The Network Police Blotter

Why Should You Enforce a Network
Security Policy?

and more . . .

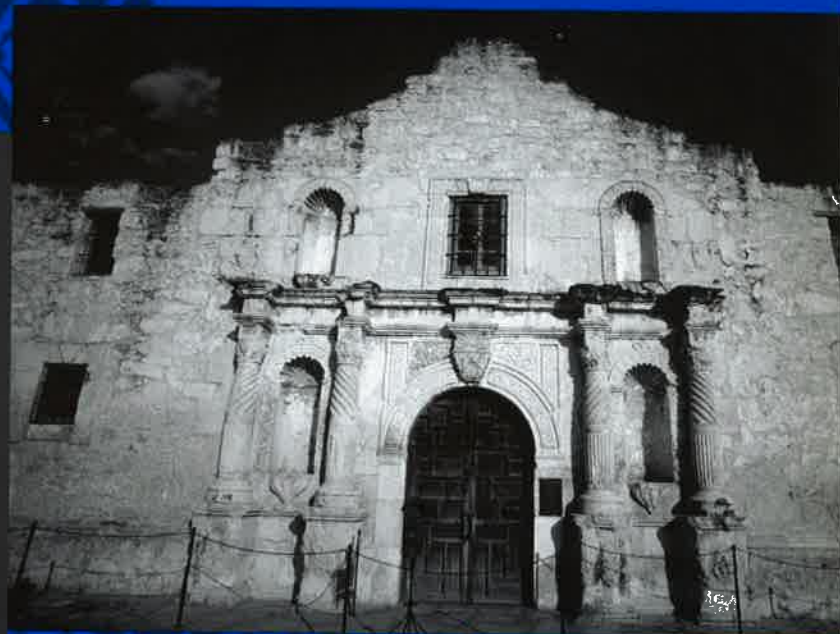
SYSADMIN

Mail Message Metering

THE WORKPLACE

Technical Interviews

and more . . .



USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

motd

by Rob Kolstad

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



<kolstad@usenix.org>

Data Overload

Boy have I been pounded with data lately. Not information, mind you, which *Websters Dictionary* cites as a synonym, but rather data. Information connotes to me a certain amount of data-processing, organization, and even distillation. Data is just raw facts, figures, random prose, and even sometimes erroneous conclusions.

I have cut back on the number of trade magazines I get. I find that once I read a few in a particular field the rest just covered the same data in varying levels of details. However, this has not really helped my overload situation.

I try to keep up with parts of the hardware world at <<http://www.eetimes.com>>. That's only 30-60 minutes/week (depending on the number of breakthroughs :)).

I also try to keep up with the software world (including open systems/open source) and boy does that eat up time! Multiple periodicals, and Web sites (even slashdot), digested newsletters (*NewsScan* is my favorite; it saves hours of reading; send a "subscribe" message to <NewsScan@NewsScan.com>). Their competitors over at *Newsbytes* do a good job, too.

The most difficult part of the problem is figuring out what information is real and what is just self-serving press releases. I am continually astounded at those who put credence in statements/data issued/uttered by people paid to do so!

Our local newspaper headline from Friday was: "Microsoft Fortress Penetrated." Fortress? That's the terminology used in their the press release.

"Isn't XYZ operating system more secure? Their literature says it is." "The ABC filesystem was created for speed, wasn't it?" "PDQ OS was designed to incorporate networking from the ground up; wouldn't that be better than UNIX's 'grafted on' networking?" So many sound bites, so little time.

"Its peak capacity is over 100 Mbits/second!" "The printer says its maximum speed is 24 pages per minute" It's hard to explain to people reading such specs that these sorts of assertions mean: "The company will give you a written guarantee that never in all of time will their device exceed this specification." Not much of a warranty, I'll aver.

And the data glut continues. Everyone who puts fingers to keyboard feels he is an expert in computing and can comment expertly on security, performance, ease-of-use, clock speeds, computer power, and data buses. I'm sorry if this sounds a bit bitter, but winnowing the chaff is ever more challenging these days as the industry jargon penetrates the popular vocabulary.

I know people who regularly receive more than 500 email messages per day. They have elaborate schemes for filing this data – often with automatic filing that takes place before they even see the message! Of course, few humans have time to digest this many messages and we often hear them wailing: "Oh, I'm buried in email," the sub-text connoting "... buried in important messages."

I know I can barely keep up and I don't even have a full-time job (at this writing) to keep me occupied for a third of each day.

I write this not to complain but to lead up to a suggested course of action. When you are fed data that is not so very useful or even data that is so slanted as to be useless, say so! Say, "Hey, this doesn't help. Here's what I need . . ."

I know marketing drives our economy, but the "lowest common denominator" is now too low for many of my needs. Lets see if we can work together to raise the bar a bit.

apropos

Is That Your Final Answer?

We bought a brand-new refrigerator-freezer last month. We plugged it in, hooked up the water, installed the shelves, put the food in, and shut the doors. We thought we were done. To our surprise, once it got cold (the manual said it takes about 12 hours to do so after it is first turned on), the freezer section began to accumulate frost. A little at first, then more as time went on. At first we thought it was part of the “bootstrap” process and once the unit reached its steady-state temperature, it would go away; but it didn’t. Finally, one morning I went to open the freezer and the door was frozen shut. I called the service department.

The customer service representative I reached was nice enough, and he seemed capable of doing most of his job, but he failed miserably in the “troubleshooting” department. He started out by asking the necessary questions: when did you buy it? where did you buy it? what is the model number? That all went well enough. Then came the troubleshooting questions: what seems to be the problem? when did it start? That’s when things started going south. I described the frost on the inside of the freezer section. I said, “It looks like the kind of ice that you see build up in a manual-defrost freezer. You know, they all used to be that way before there was the frost-free kind?” The representative proceeded with what was obviously a series of standard “freezer has frost in it” questions and comments: how thick is the frost? defrosting the freezer is recommended whenever the ice reaches 1/4-inch thickness . . . “Wait a minute!” I interrupted. “We’re talking about frost buildup in a frost-free freezer. Right?” “I’m just going through these questions,” came the reply. “Well, don’t just go through the questions. I gave you the model number; check and make sure it’s a frost-free freezer. If it is, then we don’t need to waste any time on how to manually defrost it; we’ll agree that it’s broken and we need to have a repairman out to look at it.” Surprisingly, it took me quite a while to convince him that we should really ask some of our own questions to help us get to an appropriate answer as quickly and efficiently as possible. I finally prevailed, but it wasn’t as obvious to him as it was to me.

I found myself thinking more about the conversation I had had with the customer service representative and how he didn’t seem to see the forest for the trees. It reminded me that knowing the right answer is only half the battle; knowing the right questions to ask is almost more important. For system administrators this is particularly the case. We have users of software packages, operating systems, hardware, and networks who probably don’t realize that it takes all of those layers working together to deliver the service that they’re inquiring about. It’s complicated at best. Being able to ask the right questions is essential in our line of work. Getting the key information from our users can make all the difference in our ability to troubleshoot the problem rapidly. The call I had with the appliance service representative reinforced this for me. Don’t underestimate the importance of asking the right questions.

by Tina
Darmohray

Tina Darmohray, co-editor of *login*, is a computer security and networking consultant. She was a founding member of SAGE.



<tmd@sage.org>

EDITORIAL STAFF

EDITORS

Tina Darmohray <tmd@sage.org>
Rob Kolstad <kolstad@usenix.org>

STANDARDS REPORT EDITOR

David Blackwood <dave@usenix.org>

MANAGING EDITOR

Alain Hénon <ah@usenix.org>

COPY EDITOR

Eileen Cohen

TYPESETTER

Festina Lente

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: +1 510 528 8649
FAX: +1 510 548 5738
Email: <office@usenix.org>
<conference@usenix.org>
<login@usenix.org>
WWW: <<http://www.usenix.org>>

TO RIK FARROW

FROM DR. MARTIN HERMAN

Chief, Information Access Division,
Information Technology Laboratory
<herman@nist.gov>

I read your article "Musings on Embedded Systems," that appeared in the July 2000 *;login:* and enjoyed it very much. In the article, you gave a description of Smart Spaces that was focused around the concept of a wireless island. However, the concept of Smart Spaces as generally used has a broader definition that also includes natural human-computer interaction and intelligent access to information that allows people to achieve their tasks more efficiently. The purpose of this letter is to present this alternative point of view.

I view a Smart Space as a work space embedded with computers, information appliances, and multi-modal sensors that allow people to work efficiently, together or individually, through unprecedented access to information and help from computers. Smart Spaces support sta-

letters to the editor

tionary and mobile information environments that may be connected to the Internet. A Smart Space has the following characteristics:

1. It may perceive and identify users and their actions and goals.
2. It may use speech, natural language, computer vision, and other perceptual user interfaces.
3. It provides interaction with information-rich sources.
4. It enables devices carried or worn into the space to be easily integrated with the devices present in the space. (This characteristic is closely related to your description focusing on a "wireless island.")
5. It provides extensive information-presentation capabilities.
6. It understands and anticipates user needs during task performance.
7. It provides for distributed and local collaboration, including collaboration with field personnel and mobile workers.
8. It provides improved memory and summaries of activities and deliberations for later use.

NIST is developing the Smart Space Modular Testbed based on these concepts. The testbed is currently a room which contains several video cameras, several microphones including a microphone array (for understanding speech when the speaker stands at a distance from the array), a stereo camera system, and other sensors. The testbed will demonstrate integration of multi-modal sensors with multiple applications (e.g., speech processing, speaker recognition, image/stereo processing, face/gesture recognition, information retrieval), as well as integration of wireless mobile devices and sensors.

The testbed also provides a defined middleware API for realtime data transport, a connection broker server for sensor data sources, and processing data sinks. For example, a microphone array

acquires a speech signal, reduces it to a single channel, and offers it as a data-flow. Then a speaker-identification system subscribes to the signal flow, while a speaker-dependent speech-recognition system subscribes as well. The speaker identification system then offers a real-time flow, to which the speech-recognition system subscribes. This layer makes it easier to integrate components that were not intentionally designed to work together, such as speaker identification and speech recognition systems. The data transport mechanism is also extremely lightweight and allows high throughput rates.

The goals of the project are to:

1. Develop metrics, tests, and standards to push forward underlying Smart Space technologies.
2. Develop test data and test scenarios/tasks.
3. Provide impartial, large-scale evaluations across many different systems and organizations.
4. Develop interoperability specifications and test methods.
5. Develop and distribute an integration architecture that provides the infrastructure for integrating multiple sensors, computers, and applications.
6. Provide a modular testbed for integration of component technologies. This (a) permits integration of components from different vendors, (b) permits performance evaluation of end-to-end systems, (c) permits data collection with end-to-end systems, and (d) permits testing of interoperability of components.

Please visit our Web site
<<http://www.nist.gov/smartspace>> for more information.

needles in the craystack: when machines get sick

Part 1: In Sickness and in Health: The Three Laws

In the early days of science fiction it was popular to write stories about how computers would go mad and destroy the world. To many computers users today, this prophesy apparently comes true on a daily basis, at least on a small scale. The fact that computers have taken over the world is, in a compelling sense, clear. Computers pervade our society from the kitchen to the supermarket. They are in our washing machines, our cars, our supermarket checkouts. They are responsible for monitoring and communicating the movements of a global economy, which, to a large extent, is stylized by the very computer technology it uses. They are in our televisions as well as our PCs, in video games and wristwatches. Soon we shall be wearing them and most likely even implanting them.

The fact that computers go mad and wreak destruction is also self-evident to anyone who works with them regularly. Computer programs are imperfect and they crash with astonishing regularity, destroying work and disrupting our lives. The word “mad,” which is probably more sensational than necessary, is perhaps no more than a quaint metaphor, but the basic gist of this fictional vision has a wry truth to it.

The invention and proliferation of computers in our world has not merely simplified many difficult and mundane tasks. It has also changed the way society works. In spite of the now-comical prediction by IBM’s Thomas Watson in 1945 that the world would only need five large computers, today it seems that they are as commonplace as plastic.

The fact that we rely on computers for so many of our daily activities has changed the way we live. Email is replacing pen and paper, secretaries are disappearing, we pay for items with plastic cards, we order goods from foreign countries just as easily as we do from the corner shop. We arrange our schedules according to which particular machine we have to wait for, and so on.

Because computer programs have limited capabilities, we often have to adapt ourselves to the software we depend upon. We change our work habits and reassess our wishes on the basis of what a computer can or is willing to do for us; we even work around faults (bugs) that we know to exist, bending over backwards for the machine. In a subtle but pervasive way, our use of computers controls us every bit as effectively as we are able to control them. In this sense, computers really have taken us over. It is not an evil computer intelligence that controls us, but rather something far more pernicious: we are self-styled slaves to our own almost religious use of The Machine.

Then there is the madness. Computers crash, they amplify small mistakes into huge mistakes, they do not always behave predictably, sometimes they work slowly for no apparent reason. As one popular email signature proclaims: computers allow us to make mistakes faster than any other invention in history. Sometimes they even seem stubborn in their refusal to do what we would like. They produce meaningless answers

by Mark Burgess

Mark is an associate professor at Oslo College, and is the program chair for LISA 2001.



<Mark.Burgess@iu.hioslo.no>

In this series of essays, I want to take a broader, more reflective view of what it means to build and maintain our information systems.

to apparently sensible questions. Surely the behavior of something not entirely in possession of all of its marbles.

But enough of the melodrama. What is this all about? Why do computers not just continue to work like vacuum cleaners or ovens? They are, after all, just appliances, marginally more delicate, but machines nonetheless. Why indeed do cars break down, people get sick? Why do plagues upset nature, storms wreak havoc? Are there any lessons to be learned from these parallels? Why does anything stop working the way it is supposed to? Does it have to do with foreign invaders? Bacteria and viruses? Things jammed in the works? Bits that fell off? Sabotage by enemies of the state? If so, what are these disrupters and why do they exist? Is there a corresponding gremlin that invades computer systems, causing them to fail? One thing is certain: computer viruses are not the whole story.

In this series of essays, I want to take a broader, more reflective view of what it means to build and maintain our information systems. I mean to show that many of the problems we face in running the world's computer systems are not new, even though we seem to be learning the details now for the first time. As many of my physics colleagues often complain of their research: everything has been done before; every problem has occurred and has usually been solved in some other context. There are lessons to be learned from the past, from analogous situations and also from the writings of scientists and thinkers in many contexts. It is time to revisit these ideas and reflect on their virtues, as we wobble on the springboard of the new millennium, poised to launch forth through our hand-held mobile monoliths and be reborn as masters of it all . . . or not?

Why should anyone bother to make these connections? There are two reasons for this indulgence: one is that we might actually gain a greater understanding of the specific problems we grapple with on a daily basis and see solutions through a wider perspective. We are creatures of association, and the illusion of understanding is usually enhanced by our ability to appreciate the same idea in several contexts. The other reason is that reading about things we recognize in a new context can be inspirational and lead us in new directions. "My god, it's full of wildcards!"

Naturally, no one should be looking to copy blindly from analogs in nature or history. Cheats flunk their exams, and shallow comparisons lead us only into blind alleys. Visionary monoliths are not summoned forth by beating our computers over the chassis with Lucy's favorite thighbone (though the sentiment is sometimes compelling); and, whatever Clarke and Kubrick had in mind with their 2001 monolith, it was surely more than just a mobile phone, or a CPU with the pins broken off . . . but that does not mean that we cannot take a good idea and reinvent it.

Part of the reason that we do not immediately recognize old problems when they stare us in the face is that they are embedded in a novel context: wolves in sheep's clothing, or perhaps electric sheep on our Deckard's lawn, posing as red herrings. Displaced ideas must be adapted. The key to understanding and solving associative problems is abstraction: to distill the general principles from the specific instances and retain the lessons which they convey. These may then be molded into new forms.

There are two sides to be addressed to this interaction between humans and computers. We would control our computers, and yet our computers control us. We are locked in this loop. Our preferred approach to managing information systems is somewhat haphazard, though only a little more haphazard than our general approach to managing governmental and civic functions, or our daily lives for that matter. We strive for order,

through control, but this control is based on skeletal procedures, often without a sufficient regard for the whole. Usually we wait for failure, if we even plan for it at all, and then attempt to diagnose. At the same time, our practices are defined by the computers we would like to control.

Human dependency on computers is a weakness, which makes us vulnerable to their failures. When a key system fails, the machine stops, it no longer performs its function and the loop is shattered. Not only are humans dependent on computers, but computers are dependent on humans. Recently the dangers of dependency have been considered much more seriously in connection with the possibility of computer warfare. All one needs to do to paralyze a society is to cripple its computer systems. We have built a tower of cards, based on fragile machinery, ill-equipped to face even simple threats from the environment. This mutual dependency must be broken if one is to find a more secure relationship with machines.

Interestingly, as is often the case, partial answers have already been explored by people who think up ideas for a living: science-fiction writers. During the 1940s, biochemist and science-fiction writer Isaac Asimov began to consider what kinds of problems would result from a society dependent on machines. Asimov's machines were robots, mobile computers with great physical strength, some of which had a human appearance. Asimov's future society was extremely reliant on robots. He quickly realized that machines would have the capacity to do great damage unless they were constrained with a tight leash. Asimov's answer to this problem was to endow automatons with a set of rules that curbed their behavior and prevented them from harming humans – in a sense, a theoretical immune system for humans against machines. Asimov and his editor, John Campbell Jr., together invented the Three Laws of Robotics. The laws went like this:

- A robot may not injure a human being or through inaction allow a human being to come to harm.
- A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Asimov's laws were supposed to protect humans from complex machinery in unpredictable circumstances. It was clear to Asimov that people could quickly become dependent on machines, and thus there was a need to be able to trust them implicitly. Of course, he also knew that this can never be: no matter what kinds of rules one creates, there are always unforeseen circumstances which probe for loopholes in a set of rules, and many of his robot stories were based on the mysteries arising from these loopholes.

The main threat to any system working on a set of programmed instructions is that the complexity of its surroundings tests it in every conceivable way, picking at the locks until every door is opened. Sooner or later, the system will break under the pressure. As the character in *Jurassic Park* says, "Nature will find a way." My rule of thumb is: when the complexity of its environment exceeds the complexity of a system, the system will not be completely predictable. Nevertheless, a set of rules can, clearly, greatly reduce the risk of aberrant behavior in a mechanism.

Apart from protective protocols, like Asimov's laws, which protect quality of service, security against loss of service is easily secured by a strategy that is very familiar in Nature: *redundancy*. "Redundancy is the last avenue of our survival," wrote Robert

Not only are humans dependent on computers, but computers are dependent on humans.

To understand why computers get sick, one needs to ask why they even work at all.

Silverberg of his post-biowar novel *Shadrach in The Furnace*, where the lead character is “system administrator” for Genghis Mao’s biological system. In the biological world, simple mechanisms like single-celled creatures are not well suited to survival in complex environments, unless they exist in huge numbers. The only strategy a single-celled organism has against total and final death is to reproduce very quickly and maintain large numbers, before something steps on it. If one is destroyed, there will always be a backup to continue the execution of their simple program: *copy thyself*. For computer systems, one makes backups of data to prevent loss, uses multiple servers to secure failover capacity. The general rule is that we use *parallelism* rather than serial architectures to increase the assurance of delivery. This strategy is particularly important in complex environments, because environments (by their size and nature) exert a higher level of parallelism against the system than the system is capable of producing to protect itself.

Two key strategies, then, should drive computer management: mechanisms that protect against loss of service and mechanisms that protect quality of service. The extent to which such mechanisms exist varies. Certainly no off-the-shelf computer systems have such systems as a part of their basic design. Software and hardware design is obsessed with speed and convenience, not safety and reliability. The Volvo of computer systems has yet to be sold in showrooms, though it can be put together as a kit, with a little dedication. Since the mid-1990s several researchers, including myself, have argued for (and even built) experimental systems that take on different aspects of this problem.

Computers have come a long way since the punch-card looms from which they evolved. In a compelling sense, they are now very much like primitive organisms or societies. To understand why they are so unpredictable, one needs to piece together an understanding of how they work and of the nature of their environments. This is an involved story, but as a piece of commonplace technology, it is part of our cultural heritage and so it has earned the right to a few chapters of explanation. To understand why computers get sick, one needs to ask why they even work at all. Clearly something that never changes cannot get sick. So the first step is to answer the question: how do computers change? The journey there is not a difficult one but it is surely complex and fascinating.

The next part of this story will take us into the realm of technological development and our attitudes toward it.

national laws and the net

The Good, the Bad, and the Ugly (or, Why You Should Pay Attention and Get Involved)¹

“Think globally, act locally” has been an effective motto for the environmental movement. It would also be a useful guide for national governments attempting to regulate the Net. Unfortunately, even in technologically advanced countries, governments have shown an impressive lack of understanding for the nongeographic nature of the Net. It is not the purpose of this article to provide an extensive catalog of different countries’ laws that impact the Net. Not only would such a discussion be longer than this magazine, but also it would be out of date even before it was finished. This article is intended to provide a few examples of how governments are currently dealing (or not dealing) with the Net and to encourage you to understand the ways that these national laws can impact you and how you can try to influence the formation of these laws.

There are three general ways that a national government (including its judiciary) can react to the Net: (1) recognize that some aspects of the Net extend beyond its local control and (a) cede control to an international body such as ICANN or WIPO or (b) harmonize its laws with those of other countries so that the rules are standardized (both the “good” approach); (2) ignore it (the “bad” approach); or (3) act like the Net is subject to geographic boundaries and, therefore, attempt local control and pass laws that conflict with other national laws or attempt to give extraterritorial effect to its own laws (the “ugly” approach). The majority of this article address the ugly, because that is the area that can cause the most problems and is the approach that most governments appear to be taking.

The Ugly

To visualize why the ugly approach is bad, imagine a sporting event where multiple teams compete in the same space, each with a different idea of the rules of the game and with multiple referees enforcing different, and changing, sets of rules. While such chaos might be amusing to watch for a little while, eventually it would just get frustrating for everyone. It would be expensive and inefficient.

UNITED STATES

Starting close to home, in 1999 a 15-year-old Norwegian hacker,² with the assistance of two other friends, developed a program capable of decrypting encrypted DVDs. The hacker then posted the object code of the program, called DeCSS, on his Web site. In addition to hundreds of other sites on the Web, the Web site 2600.com offered the DeCSS object code for downloading and also contained links to other sites where the code was available.

The US Digital Millennium Copyright Act³ (DMCA) states that:

No person shall . . . offer to the public, provide or otherwise traffic in any technology . . . that

by John
Nicholson

John Nicholson is an attorney in the Technology Group of the firm of Shaw Pittman in Washington, D.C. He focuses on technology outsourcing, application development and system implementation, and other technology issues.



<John.Nicholson@ShawPittman.com>

. . . it is not clear whether simply providing the relevant URL in text that could be copied and pasted into a browser is illegal.

- (A) is primarily designed or produced for the purpose of circumventing a technological measure that effectively controls access to a work protected under [the Copyright Act];
- (B) has only limited commercially significant purpose or use other than to circumvent a technological measure that effectively controls access to a work protected under [the Copyright Act]; or
- (C) is marketed by that person or another acting in concert with that person with that person's knowledge for use in circumventing a technological measure that effectively controls access to a work protected under [the Copyright Act].⁴

Various movie studios filed suit against 2600.com. The trial court granted an injunction against 2600.com and forced it to remove the DeCSS code.⁵ However, 2600.com retained its links to other sites where the code was available. Following a trial, the judge ruled that it is a violation of the DMCA for a Web site to provide a link to another Web site that provides a technical means of circumventing intellectual property protections.⁶ According to the judge, links to sites that initiate DeCSS downloads without prompting the user are violations, as well as Web sites that “display nothing more than the DeCSS code or present the user only with the choice of commencing a download of DeCSS and no other content.”⁷ For a Web site containing the DeCSS software or a link to such a Web site not to violate the DMCA, the judge said that the site must “offer a good deal of content other than DeCSS.”⁸ This case, as of this writing, is under appeal.

From the opinion, it is not clear whether simply providing the relevant URL in text that could be copied and pasted into a browser is illegal. It is also unclear whether it is also illegal to provide a link to a Web site that provides such an “illegal” link.

So what? Although 2600.com is subject to US law, based on this ruling, the Motion Picture Association of America (MPAA) has reportedly already begun sending cease-and-desist letters to other Web sites, including at least one outside the US, that provide, or provide links to, the DeCSS code.⁹ The MPAA is trying to extend US law to Web sites outside the US. In addition, for US residents, the ruling has an impact on freedom of speech, as well. So far, T-shirts listing a portion of the DeCSS code have been banned and a song with the code as the lyrics has been pulled by MP3.com.¹⁰ Finally, in an example of the effect that ownership of news providers by large companies may have, CNN, which is a subsidiary of Time/Warner (one of the plaintiffs in the lawsuit), had a news story on its Web site about the case that contained a link to a DeCSS download site. Around the time the MPAA began sending out the cease-and-desist letters, CNN removed the link (but not before the CNN page could be mirrored on several other Web sites).¹¹

How can you get involved? In US courts, parties with an interest in a case are allowed to file “friend of the court” briefs. These briefs allow you to explain to the judge issues related to the case. If you feel strongly about the potential impact of this case, talk to your company's legal department about whether it makes sense for your company to either file a “friend of the court” brief or to join in someone else's brief. Another option is for you and/or your company to contact your representatives in Congress to express your opinion.

EUROPEAN UNION

The European Commission (EC) has released a proposal that would require non-European Union (EU) businesses to register for Value Added Tax (VAT) in the EU if they make supplies of “services” to non-EU VAT-registered persons.¹² The definition of “services” includes entertainment services (such as subscription and pay-per-view TV;

streamed music and video; downloaded games, music, films and books; educational services such as online training and distance learning; and a “catch-all” list including software, data processing, and “computer services,” which includes Web-hosting, Web design, or similar services and information.

This proposal demonstrates several of the major problems that hasty or ill-considered legislation can produce:

First, the EC is attempting to apply its own laws to companies that are not physically located in areas governed by those laws. The EC proposal would impact companies that sell at least 100,000 Euros’ worth (currently around \$90,000) of “services” to EU consumers, regardless of where those companies are actually located. Thus, the EC system requires companies to identify the location of their customers in order to determine whether the company is subject to the proposal. Such identification can be difficult when consumers are downloading digitized goods. The US argues that taxation schemes like this should be subject to some kind of global regime under the auspices of the Organization for Economic Co-operation and Development (OECD).¹³

Second, the EC has proposed a system that favors one geographic location over others. The EC has proposed that non-EU companies covered by this proposal would be allowed to register in any one of the EU member states and to charge that state’s VAT. (Thus, the likely choice would be Luxembourg, since its VAT rate is currently the lowest in the EU.) This option has irritated companies registered in the other member states, since they do not have such an option.

Third, the EC has proposed a system that differentiates between e-commerce and traditional commerce. The EC has not proposed any method for harmonizing the VAT rates for physical goods and “digitized” goods – thus a book imported into a country and a book downloaded into the same country could be subject to dramatically different VAT rates.

Under the rules of the EC, a unanimous decision from all member states is required before the proposal can be implemented. The target date for implementation of the proposal is January 1, 2001.

How can you get involved? If your company could sell more than 100,000 Euros’ worth of “services” to EU consumers, discuss with your legal department and other relevant executives (accounting, sales, etc.) the potential impact this tax proposal may have on your business, including the IT costs of developing the ability to track the location of your customers. If your company decides that it wants to support or oppose this proposal, then if your company is located or does substantial business in a particular EU member state, contact the relevant government officials in that member state. If your company is in the US and does not have sufficient leverage with a particular EU member state, contact your representatives in Congress and/or the US Department of Commerce.

UNITED KINGDOM

The Regulation of Investigatory Powers Act 2000 (the RIP Act) is expected to come into force in the United Kingdom in October of this year. The RIP Act regulates the surveillance and interception of communications.

Among other things, the RIP Act contains provisions requiring all ISPs to install and maintain interception equipment for the benefit of law enforcement, to disclose encryption keys to law enforcement on demand (or face prosecution), and to keep the

The EC is attempting to apply its own laws to companies that are not physically located in areas governed by those laws.

How can you notify someone that their communication (particularly email) could be intercepted before they send it to you?

fact that the email is being intercepted and decoded a secret from other parties (or face prosecution). These provisions have resulted in several ISPs (including PSINet, Poptel, Claranet, and GreenNet) announcing their intention to remove their operations from the U.K.¹⁴

Despite the furor over the interception and encryption provisions, the RIP Act may have a larger problem. Originally, the regulations implementing the RIP Act would only allow businesses to intercept communications on their own systems for compliance and investigatory purposes.¹⁵ Among other things, originally, businesses would have needed consent from both parties to a phone call or email before intercepting it. How can you notify someone that their communication (particularly email) could be intercepted before they send it to you? In response to pressure from industry, however, the U.K. government relaxed the proposed regulations to allow businesses to monitor their networks more freely.¹⁶ Unfortunately, the official Data Protection Commissioner has drafted a code of practice under the European Human Rights Act, which recently went into effect in the U.K., that lays down strict rules for electronic monitoring and provides that a worker should have the right to a degree of trust and to work “without constantly being watched.” The code will also have the status of law, causing confusion as the courts try to decide which law is the real one.

Another problem created by this aspect of the regulations is how it impacts businesses with subsidiaries in the U.K. but whose IT systems are managed in another country with different regulations.

How can you get involved? If your company operates in the U.K., have your legal department look into the RIP Act and the European Convention on Human Rights and discuss the impact these provisions could have on your company.

FRANCE

The French have been taking a particularly aggressive approach to extending French law to the Net. In May, a French court ruled that French Net users cannot take part in online auctions except by using a government-approved auctioneer and paying French VAT.¹⁷ The French company Nart.com argued that its sales were not subject to French VAT because all online sales were handled by its New York subsidiary, where the relevant Web servers were located, and paid for in dollars to a US bank. The French court held that because Nart.com had an office in France and the sales were advertised in the French media, the sales were subject to French law and, therefore, illegal. Because Nart.com had a French parent and an office in France, this ruling is not totally unreasonable, although it should serve as a warning to companies planning to operate in France. France is holding a non-French subsidiary of a French company liable for failing to comply with French law when dealing with French users.

Also in May, however, a French court ruled that the French subsidiary of California-based Yahoo! Inc. must “make it impossible” for French users to access online sales of Nazi memorabilia, which violate French “anti-hatred” laws.^{18, 19} Yahoo!’s French site has complied by blocking access through its own site to the illegal auctions, but French users can still access Yahoo!’s US pages via other providers. Yahoo! has argued that there is no technical means of blocking French citizens who want to from seeing the Nazi memorabilia auctions. Although the French court originally demanded that Yahoo! comply by July 24, the judge has now asked a panel of experts to study the technical evidence and provide a report in November.

So what? France is attempting to hold a business in another country to a content standard defined by the French government. If the French succeed, then what is to prevent any country from legislating the content of any service available on the Web, regardless of the location of that service?

How can you get involved? If your company does business in France, these two cases should make you sit up and take notice. You should discuss with your legal department what your company needs to do from a technical standpoint. If you do not specifically do business in France, then you should pay attention to the outcome of the Yahoo! case. If the French government continues to attempt to regulate the content of non-French Web sites, then your company should consider how best to respond.

GERMANY

In Germany, courts have decided that German retail laws preclude group purchasing business models that aggregate the purchasing power of individual consumers in order to lower the final purchase price. In a particular case, a German court held that such purchasing schemes create illegal discounts and violate Germany's Retail Act and Act Against Unfair Competition.²⁰

Given this ruling, companies such as LetsBuyIt.com and Powershopping, regardless of where the company is located, must identify the location of consumers and prevent German consumers from participating in the group purchasing schemes.

So what? This is similar to the French ruling against Yahoo! discussed above. Germany is trying to prevent German users from using a service available to non-German users.

How can you get involved? The German unfair competition laws are very complicated. If your company has German customers, you should discuss with your legal department how best to respond to the German position. At the very least, your legal department should be aware of the issue.

The Bad

The Net is here to stay, and the amount of business done worldwide via e-commerce is growing rapidly. The "bad" approach to Net laws is to ignore the Net and to think that current laws will be sufficient to govern activities and crimes on the Net. A perfect example of this is the recent arrest of the author of the "Love Bug" virus.

According to various sources, the Love Bug virus cost the global economy over \$7 billion.²¹ Philippine authorities captured the person believed to be responsible for the virus, but were unable to effectively prosecute him because a hastily passed e-commerce law could not be applied retroactively. The Philippine government was forced to try to shoehorn the prosecution into a law that covers credit card fraud.²²

So what? The ability to protect the Net from criminal activity is only as good as the weakest laws around the world. However, these laws should be drafted intelligently and in harmony with those of other countries to prevent them from falling into the "ugly" category.

How can you get involved? If your company is located or does business in a country that has not kept its legal structure up to date, discuss with your legal department how your company can get involved.

NOTES

1. This article provides general information and represents the author's views. It does not constitute legal advice and should not be used or taken as legal advice relating to any specific situation.

2. Depending on which side of the DeCSS debate you are on, you might call him either a hacker or a cracker.

3. 17 USC §§1201 et seq.

4. 17 USC §1201(a)(2).

5. Preliminary Injunction, Jan. 20, 2000; Universal City Studios, Inc., 82 F. Supp.2d 211 (SDNY 2000).

6. Universal v. Reimerdes, <<http://www.nysd.uscourts.gov/rulings.htm>>, case #00-08592 (SDNY April 18, 2000).

7. Id. at 48.

8. Id.

9. See <<http://www.2600.org.au/mpaa-letter.txt>>.

10. Grice, Corey, "MP3.com yanks song with illegal DVD-hacking code" <<http://news.cnet.com/news/0-1005-2002771353.html>>, Sept. 13, 2000.

11. Tate, Ryan, "Time Warner posts a link it had banned" UpsideToday, <<http://www.upside.com/News/39a6fef00.html>>. For the screen shot of the CNN.com page, see <<http://www.scripting.com/images/cnnLinkingToDeCSSSites.gif>>.

12. Burnes, Gary, "VAT on non-EU goods", *International Internet Law Review*, pp. 46-47, Issue 6, July/August 2000.

13. Even though this is correct, the US frequently wants to have its cake and eat it, too. When it is in the interests of the US, the US frequently proposes that other countries should submit to international bodies. However, the US is generally unwilling to submit to the authority of those same international bodies.

14. Watson, Sally, "PSINet Joins ISP Stampede Over 'Snooping Bill'", <<http://www.silicon.com>>, July 21, 2000.

15. The RIP Act defines "communications" very broadly and covers telephone calls, fax messages, or emails.

16. See the DTI consultation paper at <<http://www.dti.gov.uk/cii/lbpcndoc.htm>>.

17. "France restricts Web auctions." <http://www.findarticles.com/m0ECZ/2000_May_5/61917784/p1/article.jhtml>.

[cont'd on p. 14]

Notes [cont'd]

18. Balmer, Crispian, "Court Tells Yahoo to Block Nazi Sales in France",

<<http://www.techtrn.com/techtrnnews/politicsandlaw/story/0,3685,2574671,00.html>>, May 23, 2000.

19. Rabbitte, Sonya, "Yahoo! falls under French legal eyes." <<http://www.silicon.com>>, Aug. 11, 2000.

20. *International Internet Law Review*, p. 9, Issue 6, July/Aug. 2000.

21. "Dropout to be charged for 'Love Bug'" <<http://www.zdnet.com/zdnn/stories/news/0,4586,2583821,00>>, June 14, 2000.

22. Ibid.

23. For more information about WIPO, see its home page at <<http://www.wipo.org>>.

24. <<http://arbiter.wipo.int/domains/statistics/index.html>> as of Sept. 13, 2000.

The Good

The World Intellectual Property Organization (WIPO) is an intergovernmental organization with headquarters in Geneva, Switzerland, and is an agency of the United Nations. WIPO is responsible for the promotion of the protection of intellectual property throughout the world through cooperation among its member countries, and for the administration of various multilateral treaties dealing with the legal and administrative aspects of intellectual property.²³

So far, the best example of how to deal with the global nature of the Net is the domain-name dispute-resolution process run by WIPO. Through August 2000, 1,162 domain-name dispute cases were filed with WIPO, with parties in 66 countries, and, so far, 492 decisions have been rendered, of which 98 complaints were denied, six domain-name registrations were cancelled, and 388 resulted in domain names being transferred.²⁴

So what? Although the WIPO domain-name dispute resolution process has not made everyone happy (particularly not those who lost their domain names), it is the best solution so far. WIPO has standardized the rules, which means that individuals and companies can spend fewer resources tracking multiple laws. By making the process of domain-name dispute resolution standardized, WIPO has enabled all participants in the Net to understand the rules for domain-name disputes and to predict the likely outcome of a dispute. Unlike the sporting event example earlier, this "game" has clearly defined rules that are, ideally, enforced in the same way by each referee, regardless of the geographic locations of the participants.

The Future

The "ugly" approach of attempting to apply geographically specific laws to a nongeographic service is worse than the do-nothing "bad" approach. However, neither is good. The current attempts by local governments to regulate international business means that you will have to work closely with the legal department in your company to be aware of the regulations governing your business in various countries. Given the global, nongeographic nature of the Net, it seems clear that the ideal solution is for countries either to cede control of these issues to an international body like WIPO or to harmonize their approaches so that everyone is subject to the same rules. At the very least, countries should create some system that allows companies that comply with the law in their home country to be considered in compliance with the laws of other countries.

Countries are currently struggling with the impact of the Net and how best to deal with it. If those who understand the technologies get involved, we stand a much better chance of getting laws that reflect the reality of the Net.

musings

One of the *:login:* editors suggested that it would be fitting in this December issue to look both backward and forward in time. A reasonable request, as it makes good sense not to repeat past mistakes – and to use the past as a guidepost to the future.

Even before reading that suggestion, I had opened one of my file drawers, only to find it was too stuffed to cram in a single piece of paper. I randomly grabbed an old folder that looked thick enough to provide considerable room and proceeded to glance in it before consigning it to the circular bit bucket. I was immediately amused, as well as yanked back into the not-too-distant past.

Of course, anything that occurred before 1994, the year the Internet was officially created by the US vice president, happened a really long time ago. Remember, before the appearance of Web browsers, the Internet didn't exist, or was considered “not interesting” by most people, including the editors of all commercial computer magazines.

Now go way back to 1988, and contemplate what began in that year. AT&T, which had failed to make a profit on selling UNIX or computers running UNIX systems, purchased 20% of Sun Microsystems, at the time the number-one UNIX systems vendor. Sun subsequently announced that it would migrate from using SunOS, which is based on 4.2 BSD (Berkeley Software Distribution), and move to a SVR4 base for SunOS 5 (later called Solaris).

This announcement deeply disturbed the other UNIX system hardware vendors for two reasons. First, Sun was already a powerhouse, having won the workstation marketplace and also doing very well selling UNIX-based servers (something that is still true today). The other reason was that the new fee for acquiring a source-code license, necessary for porting to architectures other than the Sparc, would be \$200,000, an unheard-of amount. The alliance of Sun with AT&T was deemed diabolical, and the Open Software Foundation was formed in 1988 to create a new operating system that would be better than SVR4 in every way.

While OSF/1 (the OS's name) would continue to support published APIs, so that it would be compatible with the older System V Release 3 interfaces, it would also have enhanced security, networking, logical-volume management, disk mirroring, and symmetric-multiprocessor capability. These goals would not be impossible, as IBM's AIX already had volume management and HP/UX had enhanced security (a package licensed from SecureWare that added ACLs and audit features). Michael Gutman, vice president of Apollo Computers (remember them?) and a moving force behind creating OSF, called the project “one of the most important events in recent computer history.”

IBM, DEC, HP, Groupe Bull, and Siemens-Nixdorf poured over \$150 million into the project, with beginning funding at \$50 million a year, and all promised to run it on their workstations and servers. Today, the only remnant of OSF/1 is True64 running on Compaq Alpha systems. The kernel, instead of being new technology developed by IBM, is the same Mach kernel being used in NextOS, as well as Apple's OS X. By the mid-'90s, both OSF and UNIX International, an organization created to oppose OSF on the public-relations front, were dead.

Free Software

What killed OSF? Was it too much money? The attempt to write an advanced operating system from scratch in less than two years? Hubris? AT&T?

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>

It wasn't Linux or GNU that brought the mighty to their knees. It was UUNET.

Nope. And although Linus Torvalds did release the first version of Linux during this time span, it wasn't Linux or GNU that brought the mighty to their knees. It was UUNET.

Excuse me, you say, UUNET? In 1988, UUNET was a small startup specializing in providing commercial email connectivity using UUCP. UUNET installed points of presence in as many large cities as it could, where users could dial up and exchange email. UUNET was not the only group doing this; PSI and a couple of the other regional NSFnet members were also beginning to provide this form of public access. Prior to this, in order to exchange email, you connected to a university on the Internet, or to a sequence of other sites that would eventually get you to a well-connected site, a process that usually involved long-distance calls. If I wanted to send email to someone in Europe, it went first to a UNIX system in Mountain View, then to Sun Microsystems, then to a server operated by AT&T in Chicago, then to Europe. In other words, I paid only for the long-distance call to Mountain View (from San Francisco), with Sun and AT&T paying the bulk of the fees. When AT&T announced that it was putting a stop to the freeloading, Rick Adams, the sysadmin at a large UUCP relay site (seismo), founded UUNET.

UUNET ran TCP/IP connections over leased lines to connect their POPs. UUNET also connected to the Internet, and soon to Europe, so it was possible to get email almost anywhere. The competitors to UUNET, past regionals, were also connected to the Internet, so while it was possible to send email between these networks, it was technically illegal, that is, against the "Acceptable Use Policy" of the Internet, which prohibited commercial access. In 1991, these companies agreed to cooperate by tying their networks together in northern Virginia, and the commercial Internet was born.

Okay, so where does free software come into this? UUNET was using SunOS on the servers that sat in the POPs and ran the banks of modems. UUNET was not very happy with Sun, because they once waited 16 months to get a bug fixed that caused panics in a serial-driver routine almost daily. So when the Computer Science Research Group at the University of California published the Net2 release containing the operating system and utilities that had been written as part of BSD, UUNET saw a great opportunity. They could perhaps create a "free" version of BSD, one that did not require a license from Unix System Laboratories (USL). This also happened in 1988 (I believe), the same time that OSF was getting funded with millions of dollars.

UUNET funded Berkeley Software Design, Inc., which initially was totally distributed and subsequently had its main office in Rob Kolstad's house in Colorado Springs. (Both have since moved). BSDi hired Bill Jolitz to write the memory manager, one of the few missing pieces in Net2 required to have a complete, working operating system for the Intel platform. Jolitz's code was published in *Dr. Dobbs's* and eventually became part of the code in the finished kernel for BSD/386, BSDi's first operating system release, as well as the core for FreeBSD and NetBSD.

Suddenly, USL's SVR4 had a serious, if poorly funded, competitor. BSDi was promising to port BSD to one of the Sparc platforms (Sun workstations and servers may share the same architecture, but not their memory management or buses), was based on the much more popular BSD (instead of System V), and would likely run faster than would SVR4 on the same hardware. Faced with real competition, USL did the only thing it could do – had its lawyers file a lawsuit against BSDi.

I was not able to find a copy of the original lawsuit on the Internet, but I did find a copy of the injunction filed against BSDi because they used the numbers 8649 in their

telephone number. Those numbers obviously present a trademark infringement, false advertising, and unfair competition (see <ftp://ftp.eecs.umich.edu/groups/lpflatt-bsd/920420.complaint>) by tiny BSDI against USL, the subsidiary of giant AT&T. USL also filed suit against the University of California, the Regents, the governor of California, and the speaker of the House, also alleging trademark infringement. You can find the friend-of-the-court (amicus) brief filed in January of 1993 at <http://www.ltg.ed.ac.uk/~richard/ftp-area/usl-vs-bsd/930107.amicus>.

For a while, things looked really grim for freedom of any form of BSD. While Linux did exist at the time, it was still years away from being a really stable OS with a good IP stack. But fortunately, USL had made several mistakes. First, they should never have taken on the State of California. USL could outspend BSDi on lawyers by a factor of 100 to 1, but with the state involved, USL lost most of that advantage. Second, USL was not keeping its agreements with UC, in that it did not include copyright notices on the code in SVR4 that came from BSD. Also, AT&T had potentially distributed the code for 32V without proper copyrights, the version of UNIX written in the late '70s at Berkeley to run on the DEC 11/780 (the VAX), essentially voiding any claims to having protected their trade secrets. And finally, the portion of the code in Net2 that they were claiming as uniquely theirs was less than 1% – 30 files – and most of the claims had to do with file, constant, and variable names.

The end came in February 1994 with a settlement. Although the terms of the settlement were secret, I did hear that the number of files that USL could actually claim contained their original code numbered four. BSDi was forced to change its phone number (which had been 800 ITS UNIX), as well as move to a new code base, 4.4 BSD (Lite). You can read the announcement at <http://download.sourceforge.net/mirrors/NetBSD/misc/release/misc/USL-lawsuit>.

The Future

So, for me, the founding of OSF was not the most important event in recent computer history. Instead, it was the David versus Goliath struggle to free BSD UNIX from the clutches of AT&T. Today, USL no longer exists, and neither does OSF. Novell had bought the license to SVR4 and the adjective UNIX, and later sold SVR4 to SCO, and gave the trademark UNIX to X/Open. There are four versions of BSD, and many version of Linux, all of which are due not just to the lawsuit, but also to the charge to open software that certainly involved UUNET, BSDI, CSRG, and a lot of courageous individuals.

This column is also supposed to look toward the future. Having just taught my favorite course twice (a survey of intrusion techniques and countermeasures), I am fully aware of what we have so far failed to create: secure operating systems. While we are getting closer, for example with OpenBSD and secure distributions of Linux, we still have a huge distance to go. Today's operating systems are large, complex, and feature-full, with Windows 2000 leading everything by a factor of ten when it comes to complexity. What we need is something different – instead of a one-size-fits-all operating system from PDA to mainframe, we need modular operating-system designs that fit into a computing fabric. Plan 9 provided an early taste of this, even if it lacked mobility features. Jini has mobility features, but is not really a computing fabric. And where is the security?

I can predict, with confidence, another email virus as damaging as ILOVEYOU for 2001. Georgi Guninski continues to find ways to use Microsoft components to execute code and/or read files (see <http://www.whitehats.org/~guninski>). Vendors will

I can predict, with confidence,
another email virus as
damaging as ILOVEYOU for
2001.

I am also predicting the rise of humanoid robots.

continue to choose features over security, simply because new features are what convince people to buy new upgrades, even if those same features make the operating system less secure.

The price of oil will continue to increase. Yes, I know this has little to do with computer-systems architecture, but it has a lot to do with the economies that support them. In September, the British lorry (truck) drivers blocked distribution of oil products for three-and-a-half days, from Monday morning until Thursday afternoon. By Wednesday, hospitals had cancelled all elective surgery, schools were closed, and most grocery stores were bare. Prime Minister Tony Blair announced that it would take over six weeks to get back to normal. And this was a three-day stoppage.

As our reliance on fossil fuels increases (bigger cars and more people worldwide driving), the fuel supply is not getting any larger. In fact, studies have shown that by 2010, the oil companies will be able to pump less oil than they do today. As the supply decreases (while the demand continues to increase), the price will inevitably increase. We need to develop alternative fuel sources now, or we will have no electricity to power our computers, much less sit in our cars for the two-hour commute.

I am also predicting the rise of humanoid robots. This will occur first in the San Francisco Bay Area, because the rising cost of housing will force anyone without a high-tech or financial job to move far away, too far to commute, leading to their replacement by robot waiters, waitresses, bartenders, cleaners, cooks, bellhops, and policemen. Well, we already have some robotic police, in the form of the cameras that take pictures of red-light runners, but that is only the beginning. Robocop, here we come.

I would love to be able to predict peace, joy, good health, and prosperity for you and everyone else in the new year. But that is for you to decide and make true.

the tclsh spot

We don't often discuss subatomic physics and systems design in the same breath, but Heisenberg described a shared problem when he pointed out that you can't observe a system without affecting it.

For example, how heavily can you monitor your servers before you spend more cycles running monitors than you spend serving? Back in the dark ages, if more than five of us used the system monitor on our PDP-11/34, the compiles we were watching would never finish.

When I want to monitor a system, I like a graphical display. A picture is worth more than a thousand words when it's presenting numeric data. On the other hand, I think windowing systems and servers go together like kim-chee and ice cream. Windowing systems chew up too many resources that an overloaded server needs. (It never seems to matter how over-spec'ed the server was – it will end up overloaded.)

So, after some thought, I decided that I had more spare network bandwidth on the LAN than I had spare CPU cycles on the server, and the obvious solution was to run a small daemon on the server to spew raw data to another box where I had the cycles to do graphic displays and fancy analysis.

Since I have even fewer spare cycles than the overloaded server, I decided to write the client/server pair in Tcl. Writing a daemon in an interpreted language to save CPU cycles may seem strange, but the Tcl-based server chews up less than 0.1% of the CPU cycles on a 300Mhz K6, and even my server can afford that much overhead.

A simplified flow for a TCP/IP server is:

1. Initialize the server.
2. Wait for a connection from a client.
3. Validate the client.
4. Converse with the client.
5. Close the connection.
6. Return to wait state.

For these simple system monitors, the conversation is one-sided – the server sends data and never listens.

The key to this client/server pair is the socket connection. The BSD socket libraries are a well-designed set of subroutines, but they aren't always the simplest things to work with. When TCP/IP socket support was added to Tcl, the BSD libraries were hidden behind new Tcl commands and the socket interface was simplified.

The loss of functionality in this simplification is that the base Tcl interpreter supports TCP sockets, but not UDP sockets. If you need more complete socket support, look at the TclDP extension.

The command for creating a socket connection is `socket`. This command can open a client socket to a server, or open a server-style socket that will wait for a connection from a client.

The syntax for opening a client-side connection is very simple:

Syntax: `socket ?options? host port`

- `socket` Open a client socket connection.
- `?options?` Options to specify the behavior of the socket.

by Clif Flynt

Clif Flynt has been a professional programmer for almost twenty years, and a Tcl advocate for the past four. He consults on Tcl/Tk and Internet applications.



<clif@clflynt.com>

A server-side socket needs to run in an asynchronous mode, handling connection requests and other events as they occur.

-myaddr addr	Defines the address (as a name or number) of the client side of the socket. This is not necessary if the client machine has only one network interface.
-myport port	Defines the port number for the server side to open. If this is not supplied, then a port is assigned at random from the available ports.
-async	Causes the socket command to return immediately, whether the connection has been completed or not.
host	The host to open a connection to. May be a name or a numeric IP address.
port	The number of the port to open a connection to on the host machine.

For most applications, you can ignore the options, and the command to open a socket is just:

```
set channel [socket $address $port]
```

The socket command can accept addresses as a numeric IP address or as a system name. For instance, you can open a socket as:

```
set channel [socket 127.0.0.1 $port]
set channel [socket www.noucorp.com $port]
set channel [socket localhost $port]
```

Once this is done, you can read and write from the channel using the Tcl gets, read, and puts commands.

A server-side socket needs to run in an asynchronous mode, handling connection requests and other events as they occur. The Tcl interpreter commonly implements asynchronous processing with call-back procedures, and the socket command is no exception to this convention.

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

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

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

```
socket -server initializeSocket $port
```

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

When tclsh opens a channel, the default is to open it as a buffered stream. Socket connections are also buffered. This can lead to some surprises when your script executes a puts but no data appears at the other end of the socket. What happened was that the data went into the buffer, but the buffer wasn't full. The socket code is waiting for the buffer to fill before any data is actually moved across the connection.

There are two ways of solving this problem:

- Flush the buffer after each write operation with a flush command.
- Change the style of buffering on the channel with the fconfigure command.

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

Syntax: *flush channelId*

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

For example:

```
set smtpSocket [socket localhost 25]
puts $smtpSocket "helo clif@noucorp.com"
flush $smtpSocket
```

The better way to solve the buffered I/O problem is to figure out what style of buffering best suits your application and configure the channel to use that buffering.

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

fconfigure Configure the behavior of a channel.
channelId The channel to modify.
?name? The name of a configuration field, which includes:

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

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

```
set smtpSocket [socket localhost 25]
fconfigure $smtpSocket -buffering line
puts $smtpSocket "helo clif@noucorp.com"
```

We now know how to configure the socket, but there is one more trick to using Tcl to create a server. The normal tclsh script is read and evaluated from top to bottom, and then the program exits. This behavior is fine for a filter-type program, but not good for a server. We really need the server to sit in an event loop and wait for connections, process data, etc.

The vwait command causes the interpreter to wait until a variable is assigned a new value. While it's waiting, it processes events.

Syntax: *vwait varName*

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

Before we can build a simple server that will send the current disk usage to its clients, we need two more Tcl commands to collect the data and schedule reports.

The `exec` command extends Tcl's functionality by letting a script use any program on the system to generate data.

The two commands that collect data and schedule the distribution are the `after` command, which lets you schedule processing to happen in the future, and the `exec` command, which will execute a command outside the Tcl interpreter and return the results (similar to the shell back-tic operator).

The `after` command has two flavors. It can pause the script execution for a time period, or it can schedule processing to happen some time period in the future. This server uses the second flavor to schedule transmitting disk usage information after 1 second (1000 milliseconds), and every 2 seconds after that.

Syntax: `after milliseconds ?script?`

after Pause Pause processing of the current script, or schedule a script to be processed in the future.

milliseconds The number of milliseconds to pause the current processing, or the number of seconds in the future to evaluate another script.

?script? If this argument is defined, this script will be evaluated after milliseconds time has elapsed.

The `exec` command extends Tcl's functionality by letting a script use any program on the system to generate data. For a UNIX disk-space monitor, we can run the `df` command. On a DOS/Windows platform, we can run `command.com /C dir` and just grab the last line.

Syntax: `exec ?-options? arg1 ?arg2...argn?`

exec Execute arguments in a subprocess.

?-options? The `exec` command supports two options:

-keepnewline Normally a trailing newline character is deleted from the program output returned by the `exec` command. If this argument is set, the trailing newline is retained.

-- Denotes the last option. All subsequent arguments will be treated as subprocess program names or arguments.

arg These arguments can be either a program name, a program argument, or a pipeline descriptor.

Finally, here is the code for a monitor program to report disk usage:

```
socket -server initializeSocket 55555
proc initializeSocket {channel addr port} {
    after 1000 sendDiskInfo $channel
}
proc sendDiskInfo {channel} {
    set info [exec df]
    puts $channel $info
    flush $channel
    after 2000 sendDiskInfo $channel
}
vwait done
```

When this server is running, you can connect to it with Telnet and see the current disk usage reported on your screen every 2 seconds.

Telnet is a good way to check that a server is working, but we can do much better than scrolling lines of text across the screen.

One simple client for this application is a text widget with a small amount of smarts about what to display:

```
# Open a client socket on the local system (for testing purposes.)
set input [socket 127.0.0.1 55555]

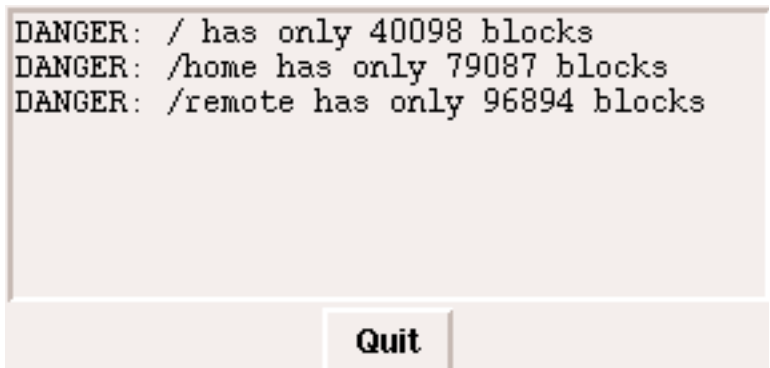
# Create a small text window.
text .t -height 8 -width 40
pack .t

# And a quit button
button .b -text "Quit" -command "exit"
pack .b
while {[gets $input line] > 0} {
    # Parse the line into a set of variables.
    foreach {name size used avail pct mount} $line {}
    # A check to see if this is a new set of df data.
    # There will be a duplicate name when we start a new dataset.
    if {[info exists Data($name)]} {
        unset Data
        .t delete 0.0 end
    }
    # Get rid of the '%' symbol in the percent usage field
    regsub "%" $pct "" pct
    # Only report if disk is more than 95 percent full.
    if {$pct > 95} {
        .t insert end "DANGER: $mount has only $avail blocks\n"
    }
    set Data($name) $input
    update
}
```

Running this client/server pair will generate a display that looks like this:

This example ignores little details like confirming which clients are allowed to retrieve info about our disk usage; the display has no historical data, and shutting down the client generates an error in the server.

I'll start filling in those holes in the next Tclsh Spot article. Until then, this code and a more full-featured client/server skeleton are available at <http://www.noucorp.com>.



```
DANGER: / has only 40098 blocks
DANGER: /home has only 79087 blocks
DANGER: /remote has only 96894 blocks
```

Quit

java performance

by Glen
McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmcl.com>



Cache-Sensitive Java Programming

One of the sayings you may have heard in connection with Java programming is “write once, run anywhere.” The idea is that Java programs are compiled into an intermediate bytecode form, and then interpreted or dynamically compiled by the Java Virtual Machine (JVM). A further aspect of this idea is that a Java application is insulated from the underlying operating system and hardware.

This approach is a “good thing” and is at least partially true in practice. For example, it’s possible to take a bytecode (.class) file produced by one Java compiler, and use that file with a JVM that comes from somewhere else.

But it’s also interesting to explore the limitations of this approach. For example, if you’re a Java programmer, do you ever need to worry about the characteristics of particular hardware that your application may run on? In this column, we’ll look at a specific example, one that involves cache-sensitive programming.

CPU Caches

As CPUs get ever faster, more attention has been focused on the CPU cache, in particular how to keep the CPU supplied with instructions and data, even though it is running faster than memory. The idea is to store frequently accessed instructions and data in a high-speed cache, instead of having to access memory. For example, suppose your application spends most of its time executing tight loops in a few key functions; it would be desirable for the instructions that make up these loops not to be fetched from memory for each loop iteration.

A specific example of a cache architecture is the Pentium III, which has a 32K level-one cache, and a 256K level-two cache, both of which run at full CPU speed. Instructions and data are accessed from the cache whenever possible, in preference to memory. A typical cache replacement algorithm brings in 32 bytes (a “cache line fill”), aligned on a 32-byte boundary.

An Example

What does knowledge of CPU caches have to do with Java programming? Suppose you’ve got an application that uses a big array, and you’re accessing array elements either sequentially or else in random order. A demo program that models this situation looks like this:

```
import java.util.Random;

public class CacheDemo {
    // size of array
    static final int N = 1*1024*1024;

    // random number generator with fixed seed
    static Random rn = new Random(0);

    // array of bytes to sum up
    static byte vec[] = new byte[N];

    // index giving order of access to vec elements
    static int index[] = new int[N];
```



```

// fill the array with random bytes
static void fill() {
    for (int i = 0; i < N; i++) {
        vec[i] = (byte)(rn.nextInt() & 0x7f);
        index[i] = i;
    }
}

// sum the values of the array,
// accessing elements according to the order in index
static int getsum() {
    int sum = 0;
    for (int i = 0; i < N; i++)
        sum += vec[index[i]];
    return sum;
}

// shuffle the elements of the index
static void shuffle() {
    for (int i = N - 1; i >= 1; i--) {
        int j = (int)(rn.nextFloat() * (i + 1));
        int t = index[i];
        index[i] = index[j];
        index[j] = t;
    }
}

// driver
public static void main(String args[]) {
    long start, elapsed1, elapsed2;
    int sum1 = 0, sum2 = 0;

    fill();

    // sum the array elements, accessing elements in sequential order
    start = System.currentTimeMillis();
    for (int i = 1; i <= 25; i++)
        sum1 = getsum();
    elapsed1 = System.currentTimeMillis() - start;

    // sum the array elements, accessing elements in random order
    shuffle();
    start = System.currentTimeMillis();
    for (int i = 1; i <= 25; i++)
        sum2 = getsum();
    elapsed2 = System.currentTimeMillis() - start;

    // sanity check
    if (sum1 != sum2)
        System.err.println("sum1 != sum2");

    // display results
    System.out.println("sequential order = " + elapsed1);
    System.out.println("random order = " + elapsed2);
}
}

```

The program creates an array and fills it with random values. Then another array is created to contain indices into the first array. We then do two timings, each of which goes through the array and adds up all of its elements. The first timing accesses the

array in sequential order, the second in random order, using the index array to control the access order.

Timing Results

I tried this program on a 300MHz Pentium II, using a couple of different Java compilers, and the results in each case show that accessing array elements in random order is about 2.5 times slower than accessing them in sequential order.

It's possible to create other similar programs in Java or C that show the same behavior. For example, if you take a C array and sum its elements by accessing, say, every 128th element, in a sequence like this:

```
0, 128, 256, ...  
1, 129, 257, ...  
2, 130, 258, ...
```

you are likely to observe similar timing behavior.

These results are consistent with the way hardware caches work. If you access elements sequentially or access elements that are close together in memory at about the same time, then it's more likely that the elements will be in the cache, because of the way that memory locations are grouped together.

It's possible that you may observe very different behavior. For example, some old microprocessors might not have a cache, or their cache may run relatively slowly. There are also CPUs like the Pentium Xeon with caches as large as 2MB.

Should You Care?

Should you worry about cache-sensitive programming? Most of the time, no. A direct analogy can be made to programming in assembly language versus high-level languages like C. A skilled assembly-language programmer can typically turn out code that is faster than C code, but there's a cost to doing so – the programmer has to manage much more detail, and such programming is time-consuming and error-prone.

The same is true with Java programming. If you spend a lot of time worrying about cache behavior on the target machine, you may end up writing code that's convoluted or tricky to understand. And with Java code, which is supposed to be widely portable, picking a specific target architecture in the first place may be hard to do.

Having said this, understanding how CPU caches work is useful, and sometimes important in specific applications. The trend in hardware seems to be toward placing more emphasis on the cache, suggesting that this area will become more important as time goes by.

Further Reading

Jon Bentley discusses cache-sensitive programming in his *Programming Pearls* (Addison-Wesley, 2000). The Web site for the book is <http://www.programmingpearls.com>.

using java

Enterprise Java Beans: An Overview

It goes without saying that distributed applications are here to stay. One reason for this is that hardware and software advances have made it possible for computers to talk to one another across a network relatively more easily than perhaps ten years ago. Whereas hardware has permitted connectivity of a heterogeneous arrangement of machines on a network, writing the software has been a more arduous endeavor.

The enabling capability that facilitates the running of distributed applications is the Remote Procedure Call (RPC), a network service whereby a remote client can make a procedure call that may be executed on a remote server.

Although RPC was a major breakthrough in permitting networks of machines to run distributed applications (filesystems are one example), the Application Programmer's Interface (API) – was not simple to master, affecting the ability to write robust and extensible applications in reasonable time.

In more recent times, RPC has been adopted as the transport mechanism for the Component Object Model (COM) and Distributed COM (DCOM), as well as the Common Request Broker Architecture (CORBA) and Remote Method Invocation (RMI).

In this article I present a brief overview of the various object technologies and then a summary of Enterprise Java Beans (EJB). In future articles I will provide more detailed examples on how to write an EJB application.

The Enterprise Environment

The term “enterprise” is ubiquitous in today's computing jargon. It typically means that the environment in question has to contend with and accommodate the changing needs of the organization deploying it.

For instance, in an Enterprise Environment it is typical to have to deal with:

- legacy systems
- decentralized IT decision making
- commercial products
- home-grown applications
- new acquisitions.

The main thrust of an enterprise environment is the pressure to extend, integrate, and evolve functionality over time.

Most if not all of these capabilities are driven by the need to get software to market on time or ahead of competitors, and also to do it at lower cost and higher efficiency and with more stability.

THE DISTRIBUTED COMPONENT OBJECT MODEL (DCOM)

The first technology to facilitate the deployment of an “enterprise” solution was DCOM from Microsoft. More specifically, DCOM permitted the development of small pieces of code called “components” that could be connected to other components of the same type using well-known APIs. The windows “registry” was the central repository for information regarding DCOM “servers,” and client applications were able to query the registry for the location of the server and so avail itself of its services. This technology

by Prithvi Rao

Prithvi Rao is the co-founder of KiwiLabs, which specializes in software engineering methodology and Java/CORBA training. He has also worked on the development of the MACH OS and a real-time version of MACH. He is an adjunct faculty at Carnegie Mellon and teaches in the Heinz School of Public Policy and Management.



<prithvi+@ux4.sp.cs.cmu.edu>

Another way of looking at this is that CORBA is the rest of the world's answer to DCOM.

was originally restricted to Windows-based platforms, but recently it has been made available on UNIX-based machines. Although DCOM did not strictly fit our definition of an enterprise solution, it approaches it more so today.

THE COMMON REQUEST BROKER ARCHITECTURE (CORBA)

The CORBA movement was born shortly after the advent of COM and DCOM from Microsoft. Another way of looking at this is that CORBA is the rest of the world's answer to DCOM. The CORBA movement involved several hundred organizations and it resulted in a CORBA specification. Various implementations of this specification are available today. The key concepts behind it were:

- Protect software that we write today and allow a choice of API.
- Make software interoperate; we cannot rewrite most of it.
- Concentrate on control and invocation interfaces.
- Expect technology to evolve.

It is fair to say that CORBA has gained popularity because it permits a heterogeneous arrangement of hardware and software and is more aligned with the enterprise solution than perhaps DCOM.

REMOTE METHOD INVOCATION (RMI)

The development of Java from Sun Microsystems brought yet another distributed-object technology in the form of RMI, which is also based on RPC.

The main difference between Java/RMI and the other two distributed-object technologies is that RMI requires that the server and the client be written in Java. More accurately, they must present Java calling semantics to each other. This means that although the server and client can be written in a “native” language, they must be “wrapped” in Java.

We can argue that Java also fits into our definition of an enterprise solution because the Java Virtual Machine (JVM) has been ported to a wide range of platforms.

Java Beans

The Java answer to components was the “Java Bean.” The Java Bean is a Java program that conforms to well-defined rules regarding how it should be written. Another very simplistic way of looking at it is that it conforms to a “programming standard” of sorts. I will present a more detailed discussion on Java Beans in a future article.

An important aspect of Java Beans is that it permits software developers to write small components much the same as in COM and DCOM and then build larger pieces of software using the smaller building blocks. Eventually a complete application can be constructed using this method. Typically, graphical user interfaces lend themselves well to this strategy.

ENTERPRISE JAVA BEANS (EJB)

Enterprise Java Beans form part of a standard component system for application servers. Some features of EJB technology are:

- It is an interface standard for which vendors provide implementations.
- It has no relation with client Java Beans.
- It has gained wide acceptance in industry (35 vendors provide EJB-compliant interfaces).

The EJB package is part of the Java 2 Enterprise Edition (J2EE), which was announced in April 1997. EJB 1.0 was released in March 1998. The specification was defined by Sun in partnership with BEA and Oracle.

Some objectives were to:

- standardize Java application servers
- enable reuse of business components without access to source code
- make component and application development easy.

The EJB specification specifies the interface between:

- a client and an EJB
- an EJB and the EJB container.

An EJB application server includes:

- the server
- development and management tools
- standard infrastructure services such as RMI, security services, naming services, transaction services, database connectivity, and messaging services.

Summary

I have presented a brief overview of the common distributed-object technologies. The development of DCOM, CORBA, and RMI were all driven (in part) by the desire to support the notion of “enterprise” solutions. Enterprise Java Beans, arguably, bring new meaning to this expression. It remains to be seen whether EJB technology will live up to this reputation.

the network police blotter

by Marcus J.
Ranum

Marcus J. Ranum is CTO of Network Flight Recorder, Inc. He's the author of several security products and of a book on computer security (with Dan Geer and Avi Rubin) and is a part-time sysadmin.



<mjr@nfr.net>

The first year of the twenty-first century is ending. I guess that's not a particularly big deal unless you're an Arthur Clarke fan. Did you notice that in 2001, the HAL 9000 computer suffered a security incident, when it was apparently subverted by an alien trojan-horse program? I guess firewalls don't work in the sci-fi future. Then again, I'm not convinced anymore that they work now. My impression of computer security in sci-fi is that it's either ignored as if it's a solved problem or it's a plot device in which the aliens'/bad guys'/good guys' systems are penetrated using an unsophisticated trick like guessing a password. With a bit of luck, computer security will somehow improve without requiring a complete parallel upgrade of Homo Sapiens V1.0.

I've long maintained *Ranum's Law*, "You can't solve social problems with software" but never projected into the future far enough. The implication is that computer-security problems will remain the same as they are today, no matter what we do – because people, not computers, cause problems. Have you ever seen a computer hack itself? It takes a human to do something that stupid.

In order to make the next great leap in computer security, we need to change our attitudes and address the problem socially, not technologically. I've been really happy to see the beginnings of a sea change in opinion about hacking – recently, a number of companies have been jumping loudly on the "we don't hire hackers" bandwagon. Next, I'm guessing we'll see wider recognition of the fact that a lot of the "grey hat" hacking going on consists either of ego-driven attempts to count coup on unpopular vendors or stealthy attempts at marketing security services. Getting a clear picture of things is important to making progress – especially with social problems.

Even the best technology is not going to help unless it's wisely and correctly used. As I was pondering computer security in sci-fi, I had an amusing mental picture of Capt. James T. Kirk telling the ship's computer to do something and having the computer reply, "Password." We've known passwords were an obsolete technology for a very long time now, but we're still using them – so why will the future be any different? It'll only be different if we make it so.

Are there any good things on the horizon in the short term? Well, I'm seeing one trend that I like, and a recent item supporting it from none other than Microsoft. The current trend of security is "penetrate and patch" – find a bug, fix a bug, find a bug, fix a bug. This is also known as "kludging your way into heaven," and it would work except that heaven doesn't approve of kludgy code. A few years ago I was drinking too much beer in some hotel bar at some conference, and predicted that eventually we'd see software that would update itself automatically: instead of reading BugTraq and seeing that (oh, joy!) you have to rush and upgrade your Web server because 10,000 script kiddies were just given a tool that breaks it, you wake up in the morning with a nice email from your Web server informing you that last night it upgraded itself and it hopes you're enjoying your coffee. The antivirus software vendors figured this out a while ago – having your defense system update itself automatically is a good thing. Why can't my operating system do that also? And my Web server? And my firewall? In fact, that's what this new thing from Microsoft does: it checks to make sure your copy of your Web-server software is up-to-date with security patches. It's a start, anyhow. It's called HFCCheck. If it works, I bet you we'll see more tools like it.

One of the biggest problems with patches is getting the user to actually install them. I'm hoping that eventually they won't have to. I'd like to be able to install a piece of software and, when I install it, tell it, "You're mission-critical. Notify me immediately if you need to be updated but don't do anything until I tell you to." Or, "You're security-critical. If you need to be updated, just do it automatically or shut yourself down and call for help." There are downsides to this concept, of course. One of them is that it further legitimizes the "penetrate and patch" model – it just speeds up our ability to patch. The other is that it may further reduce the role of software testing. If a vendor can just throw patches out one after another, what's the point in getting a release "right" anymore? I suspect software testing is already a casualty of "Internet time," though. With the full-disclosure crowd insisting on patches with a subweek turnaround, it's no wonder that everyone is being forced to run beta-test software. Perhaps our future piece of software won't simply be told, "upgrade yourself if you need to." We might tell it, "upgrade yourself only to improve reliability; don't upgrade yourself to add features without asking me first; upgrade or halt to fix security flaws." This would move us to a model of software development in which everyone is running the "code du jour." Is this a fair price to pay? I think we're already paying the price but aren't yet reaping the benefits.

Incidentally, I believe the big breakthrough in self-upgrading software will come when some business visionary points out that the self-upgrade cycle represents a perfect opportunity for a vendor to favorably "touch" its customers at a regular interval. If the whole Application Service Provider model works out, we may even see self-upgrading software as an eventual implementation of "rental" software – you get to run the latest and greatest version of whatever it is as long as you're paying a subscription fee monthly or annually. Microsoft has already pioneered this approach with its developer tools, which presently self-upgrade in the form of a steady stream of CD-ROMs sent in the mail. Indeed, the operating systems of the future might offer load-on-demand application services. Imagine for a second if your operating system came on a 1.4M floppy disk and installed itself in 20 seconds. Then, the first time you try to send email it presents you with a list of mail systems, their subscription prices, and so forth, and lets you pick and go. Software might look like signing up for satellite TV: you'd buy a number of "points" with your platform package that could be "spent" to subscribe to a browser application, a word processor, a mailer, a security audit service, and perhaps an archival backup service. This implies that system administration will need to be a "solved" problem within the next ten years (no more Windows installation questions, please) – is anyone out there listening?

Some Feedback

My recent articles about hacking and full disclosure have certainly touched a few nerves. I asked for feedback/rebuttals and have gotten more than I can print or even paraphrase here. About 70% of the messages I got were actually supportive; it's good to know I'm not completely in la-la land. So, some feedback.

Jeff Root writes:

The short version: what you describe is a fantasy version of the future. It will never come to pass.

The longer version: what you propose (i.e.: launching lawyers against full-disclosure sites) is emotionally satisfying, but is analogous to ridding the kitchen of cockroach-

One of the biggest problems with patches is getting the user to actually install them.

es by turning on the lights. It appears to work; there are never any visible critters, but in fact their actual number does not decrease.

Analogies are always dangerous, since they can be extended beyond their intended bounds, often with ridiculous results. What I propose is actually more analogous to putting all the food away in the kitchen when it's not in use, and spraying a little insecticide in all the obvious roach-breeding spots. It appears to work, there are never any visible critters except for a few dead ones, and their actual number does decrease. Sometimes it decreases enough that the roach population becomes manageable.

A. Nonymous writes:

We must know our enemy. This is a fact of life everytime good guys try to combat bad guys/things: in war, medicine, biology, earthquake . . . I learned much more on BugTraq than in all vendor bulletins combined. We (white hats) must have access to as much information as possible in order to better prepare our defenses. This implies free access to technical descriptions of the holes, probably also exploits to test our defenses, and freedom to reverse-engineer the programs that we use so that we can protect them before the vendors do . . .

“Know your enemy” is an ancient maxim of warfare – I’m sure Sun Tzu wasn’t the first person to say it. But, in wartime, there are stiff penalties for “aiding and abetting” the enemy – which, in my opinion, is what a lot of “grey hats” are doing when they release tools to script kiddies. We “white hats” must have access to as much information as we need to strengthen our systems – as far as I am concerned it is enough that a vendor tells me to “upgrade XYZ pronto!” I revere curiosity as the root of all human learning, but sometimes it’s best left unsatisfied. We white hats already know about buffer overruns and how to avoid writing them into our code – it’s not intellectually stimulating to be drowned in a sea of reports of buffer overruns in versions of software you don’t run.

A. Nonymous continues:

In crimes (and cybercrimes are primarily crimes) the intention is more important than the objects used. Some people carry guns but don’t kill; some commit murders with ropes or knives but we don’t ban these objects. Releasing an exploit program is like selling a knife: it’s morally OK and the real difference is in the mind of the end user.

“Toolz don’t hack Web sites, script kiddies do!” – perhaps the NRA will adopt the slogan one day, but I doubt it. There are two important issues at play here:

- what is legal
- what is moral.

Lawyers exist to address (some might say “muddy”) the first of those. The second is one that individuals must resolve in their own hearts with whatever guidance they deem appropriate. I believe that there are some serious ethical lapses in the security field today – I don’t believe that what’s going on is necessarily illegal. Yet.

In the parts of the world that have not fallen into anarchy, the ownership and distribution of guns are regulated. You’re right that ropes aren’t. They aren’t for two very good reasons:

- It is much easier to kill someone with a gun than a rope or a knife.
- Ropes and knives have a wider variety of benign uses than guns.

For the record, I am a gun owner and enjoy shooting paper targets. I'm not happy about having to register my guns; unfortunately, a lot of bad apples have already spoiled that barrel for the rest of us. I'm *very* unhappy that irresponsible individuals have made suspects out of those of us who do act responsibly. Part of my sense of responsibility includes recognizing that if some kid got his hands on one of my guns and did something stupid with it, I'd feel terrible (and probably spend a lot of money on legal bills). Consequently I'm very careful to keep them unloaded and locked in a very secure storage facility. See any parallels here?

Societies choose to regulate such things to a greater or lesser degree based on the trade-off of utility to lethality in the hands of an unpremeditated or inexperienced killer. I think that society may make a similar determination with hacking tools. Releasing an exploit program is not like selling a knife; it's like handing out loaded submachine guns and claiming that the end result is not your responsibility. I don't buy that, and neither would most juries.

By the way, in many parts of the US you must be over 18 to buy certain configurations of knives. The seller can get in a lot of trouble for selling a double-edged combat boot knife to a minor. You can also get in a lot of trouble for carrying one on your person, whether you intend to use it or not; be especially careful about airport metal detectors.

With power comes responsibility. Many of the nifty-keen Internet technologies we build are powerful tools that will help shape the future of global information. Those of us who build those tools – or build tools that might be harmful if abused – must accept responsibility for the consequences of our actions.

Harald Nordgard-Hansen writes:

I often find one thing that really annoys me. That is all the vendors that tend to brush away or downplay the severity of security holes with a statement along the lines of “no exploits exist for this hole” or “has not been observed in the wild.” There seems to be an implicit assumption that until someone writes an exploit and distributes it, the hole is not really serious and there is no reason to allocate resources in the organization to fix it.

Downplaying the severity of a hole is a double-edged sword, indeed. On one hand, if vendor make the hole sound minor, nobody may install the patch. On the other hand, if they make the hole sound major, they may trigger a panic. Remember back in the days when something like the Michaelangelo virus was a big deal? I suspect that people are becoming so jaded by the relentless flood of bugs that it'll be hard to raise their blood pressure anymore. I wonder sometimes if the vendors are afraid that someday they may be held liable (like, say, for example, tire manufacturers?) for failures in their products that result in end-user damage. In fact, I'm *sure* they're afraid of the idea, which is one reason they're trying to head the threat off with UCITA.

This is one of the big changes that has to happen. Vendors have to own responsibility for taking security bugs seriously and fixing them in an appropriate and safe manner. Some of them have gotten away with dodging that responsibility for a long time. I think that era has come to an end, though.

The issue of making vendors take a bug seriously until there's an exploit in the wild is a serious one. By now it should be obvious that once a bug is found, someone's going to do a press release about it, so it's just a matter of time until there's an exploit. However, I don't think responsible security practice is to *force* their hand unduly by threatening them with an exploit, the way many “grey hats” do. Let's suppose someone discovered

Releasing an exploit program is not like selling a knife; it's like handing out loaded submachine guns and claiming that the end result is not your responsibility.

that a certain brand of tires failed catastrophically if they were driven over marshmallows. Responsible practice would not be to scatter marshmallows all over the streets “to get people to take the problem seriously.”

Adam Shostack writes:

my system got hacked
lamerz get slammed in juvenile court
like fish in the sea

Last time I checked, CNN said that “mfiaboy” (allegedly the “brains” behind the denial-of-service attacks against amazon.com and CNN) had 64 new charges being levelled at him by prosecutors. Think of it as evolution in action.

Up Next

Next column I promise I’ll try to lighten up a little bit. For the next contest, I’ll accept nominations for “practicality prize” winners. These are to be real-life, incredibly great things that are overlooked and perhaps underappreciated. Let’s give them their day of glory! Best suggestions will get a nifty windbreaker. Some sample favorite practical things I love: those nitrogen doodads in the cans of Guinness – what a brilliant idea. And those Velcro cable ties. And . . . Email to <mjr@nfr.net>; Subject: “practicality prize.”

why should you enforce a network security policy?

The purpose of a network security policy is to protect all proprietary and confidential information assets from unauthorized access, disclosure, modification, misuse, or destruction. In any organization, regardless of size, such a policy is a requirement for the secure functioning of the environment. A policy ensures that the necessary protection is enabled for essential business activities to be conducted. A good policy establishes the necessary procedures to enable productivity, security, accuracy, and availability of data and resources.

Organizations frequently mistake procedures for a policy. A policy is the vehicle that implements all security procedures. A policy can define what the parameters are for the organization's security stance and what procedures and low-level details are necessary to enforce it.

No Policy = Get Hacked

This is a pretty bold statement, but all organizational security is derived from policy enforcement or lack thereof. Enforcement of policy is the major problem in most organizations. Too general a policy will allow procedures or actions that should not be permitted. Too restrictive a policy will cause users to break or ignore it. A good policy is useless if management does not support it and enforce its usage. Management can actually be the worst enemy of a good policy. If management does not understand the need for a security policy, you might as well open the doors and welcome the hackers in. This article describes how to set up a usable network security policy that is easily enforceable and can be scaled to support different types of organizations.

When defining a network security policy, the first question that has to be answered is, "To whom does this apply?" The correct answer is everyone. There are many disparate groups in any organization, but one of the things they all have in common is the need for security. It does not matter what type of company it is; one network security policy needs to be applied across the board and adhered to for any kind of security to be in place.

The first group within the organization that the policy has to address is the user community. As power is increasingly placed in the hands of users, their responsibility for security is increased. The policy must spell out what their responsibilities are for securing the organization.

The next groups that need to be addressed by the policy are the system administrators and network administrators. The bulk of security issues concerns the system administrators. As with users, the network security policy must address their roles and responsibilities for security.

Administrators frequently encounter situations in which breaking the policy would provide a convenient and practical way to solve some problem quickly. In any production environment, this may seem a necessary thing to do. When it becomes habit, however, the reason for the policy becomes lost. The security of the network can be compromised very quickly if expedient rather than secure methods are used that fall outside the network security policy. The policy must be flexible enough to address emergency situations yet keep the environment secure.

by Gary Bahadur

Gary is the Chief Information Officer for Foundstone Inc <www.foundstone.com>. He has been performing security assessments for the past 6 years and teaching security training courses for the past 2 years.



<gary.bahadur@foundstone.com>

Test environments are notorious for security weaknesses that can compromise production environments.

One of the great pitfalls of many environments with security policies occurs when administrators set up test environments that do not adhere to the policy. Test environments are notorious for security weaknesses that can compromise production environments.

The third group that is crucial to the success of a network security policy is management. Without management's full support of and adherence to the policy, it will fail. Management that skirts the policy for convenience sets the example for the rest of the organization. A top-down approach to security is necessary to gain buy-in by all users in the organization.

Policy Definition

What makes a good policy? In a UNIX environment, the qualities that make up a good network security policy can also be applied to most other environments. A general policy that can be applied to various network environments must describe all aspects of the environments, from system usage to the configuration of password controls, in detailed procedural documents. As a general rule, a network security policy has to address the following areas:

ROLES AND RESPONSIBILITIES

A key aspect of the security policy is the definition of principal roles and responsibilities. Positions such as Security Office, System Administrator, Database Administrator, and Information Security Committees have to be defined by the policy.

SYSTEM USAGE

- An awareness of the importance of security must be disseminated to all users in the organization. Users need to learn system usages' security issues in some detail.
- Users must have authority to utilize the system.
- That authorization must be given by management.
- Management must also define the manner in which the systems can be used.
- The proper use of email has to be specified.
- Users may not use the system for unauthorized activities such as setting up bulletin boards or warez servers.
- Moves between departments or levels usually change user access privileges.
- Information defining the proper usage of the system upon login should be determined in the policy.
- Time restrictions such as automatic logoff, restricted night access, and weekend access should be specified by the policy based on job function.
- The ability of third parties to access internal systems should be clearly defined in the policy. How third parties connect, when they can connect, and what systems and data they have access to should be addressed.
- Methods of connecting to the internal and external systems, the encryption methods that are to be used, and authentication methods need to be clearly described. Access paths can cause numerous problems and should be clearly defined.

CONFIDENTIALITY

- The levels of confidentiality of information must be defined, and user access to each level must be authorized. Each level should be labelled, e.g, Secret, Confidential, and Public.
- Data networks must be secured through the use of an appropriate combination of controls, including authentication, validation, and logging. Encryption standards for the transmission of data must be implemented to keep it confidential.

Password strength is a key security measure in any organization.

- Encryption standards must be established. Encryption can be used for documents both publicly and internally transmitted.
- The policy must state who can disseminate information, what type of information can be disclosed, and where.
- The privacy of user information and company information must be established, in particular, management has to define what privacy rights users will have.
- The policy must describe how user activity is tracked through log files and when management can use logs to follow up on suspicious activity.
- The privacy of email must be determined, and users notified when email can be read by management and what their rights are to personal email.
- Information should be available only on a need-to-know basis.

PASSWORD/ACCOUNT MANAGEMENT

- Password strength is a key security measure in any organization. Standard measures of password strength include alphanumeric, 6+ characters in length, nondictionary words, and special characters.
- Passwords must be changed periodically and should not be written down or shared.
- Procedures should be in place to authorize users to obtain an account and password and to provide for the secure dissemination of the account information.
- Secure initial passwords should be set. Forced password changes should be implemented if the system cannot do it automatically.
- Accounts should follow a predefined scheme for creation, modification, and deletion. Accounts should be disabled when not active.
- Account review should be done periodically to validate all users.
- Unsuccessful account login attempts should be investigated and reported.
- Newer technologies such as token authentication should be explored to increase password security where possible.
- A process for securing vendor software to strengthen weak passwords should be in place.

APPLICATION USAGE

- Define how encryption should be used for critical application connections or data transfers.
- Virus protection for both applications and the operating system must be established. Updates of the virus-checking software must be timely and efficient. Viruses targeted at email and Web usage are on the rise and should be addressed.
- Software installations by both users and administrators should be authorized and done according to license agreements. The policy should deal with the issue of copyright agreements.
- Application design and usage should follow set security procedures that provide secure operations.
- Proper usages of applications and data for business and nonbusiness purposes should be defined.
- User access to applications should be based on necessity and authorized access. The policy should detail who can access what types of applications and data.
- Developer access to production applications and data should be closely monitored, and records reviewed to restrict the possibility of internal security breaches or unauthorized application changes.

Remote access through either Internet connections or dial-up connections should be monitored and audited.

INTERNET USAGE

- The business purpose of Internet usage should be clearly defined for both users and administrators. Users should sign an authorization form outlining the rules of proper Internet access..
- Sending data across the Internet will usually be in clear text. The policy should define what types of data can be transferred in the clear and which data should be encrypted.
- Email from and to the Internet should be closely monitored, checked for viruses, and authorized.
- The policy should describe how system administrators monitor Internet usage and track activity, whether authorized or unauthorized.
- All Internet services should be authorized. Administrators and users should not be allowed to start or stop new services without authorization. The policy should address how services are authorized, used, and monitored. Both inbound and out-bound servers need to be restricted.

BACKUP AND RECOVERY OF DATA

- Data backup and recovery parameters should be set forth in the policy. General outlines such as how often and who performs backup and recovery should be described. Detailed procedures for operating systems and application data can be handled by individual procedural documents.
- User backup of data should be addressed in the policy. Users often have data on desktops that does not get backed up on a regular basis.

DEVELOPMENT PROCESS

- The development process should be defined to follow set procedures. Change-control procedures have to be defined as part of the development process.
- Security should be determined before an application is completed. The details have to be developed in the procedures of the development life cycle.

REPORTING/AUDITING

- Security violations or concerns need to be defined, and the process or chain of command needs to be defined. Problems can include viruses, hacker attempts, system-administration errors, or internal employee problems.
- Audits should address all system areas to report problems such as invalid login attempts, invalid access to production data, hacker activity, or system errors.
- Remote access through either Internet connections or dial-up connections should be monitored and audited. Review of access attempts should be monitored and recorded.
- Log-file collection, backup, and security should be described by the policy. Secure log-server usage should be determined in the policy and the procedures defined by the operating-system documentation.
- The usage of auditing tools and the process of when and how to use them should be part of the policy.
- A process for management review of audit logs and system logs should be determined. Administrator review of the logs should not be the final review of system activity.
- A risk analysis should be performed to determine what systems need to be audited and what security has to be in place, followed by a determination of where to allocate resources based on a risk-classification level.

OPERATING SYSTEM

- New services must be closely scrutinized and authorized before being implemented. Services should be based on a business justification. An emergency process should be in place to handle exceptions to the process. The policy does not need to cover each service, just the process for authorization, usage, monitoring, and reporting.
- Users with access to the operating system must be provided with security training specific to operating-system concerns. User training can offset much low-level vulnerability that can compromise a system.
- Installing up-to-date patches and upgrades to the operating system is critical to the security of the organization. Changes to the operating system must be accomplished in a timely and efficient manner.
- Periodically searching the operating system for suspicious files or modification of critical files should be required. The process of file-integrity checking is key to understanding changes in the environment. Monitoring changes is closely tied to installing and utilizing real-time monitoring programs to secure the operating system. The details are left to procedural documents, but the policy needs to determine where and when monitoring of the operating system needs to occur.
- The policy should determine how trust relationships between systems and external connections are secured.
- Reporting at the operating-system level should be determined for administrator usage, activity of users, and potential threats and exception reports of errors or hacker activity.
- Utilization of system monitoring, scanning, and reporting tools should be defined by the policy. Procedural documents can detail which software needs to be used.

Periodically searching the operating system for suspicious files or modification of critical files should be required.

Policy Do's

- Management support of the policy is critical.
- Make the policy general enough to cover all functions of the network.
- Periodically update the policy for changes in the environment .
- Assign enforcement of the policy to a group such as the audit group.
- Require everyone to sign off on the policy when they start working.
- Make the policy easily accessible on the corporate intranet.

Policy Don'ts

- Do not make the policy too detailed or restrictive. Too many details in an overall network policy can cause confusion or backlash in implementing it.
- Do not make exceptions to the policy for individuals or groups.
- Do not tie the policy to specific computer applications or software.

Why Policies Fail

It bears repeating: The number-one reason for policy failure is lack of management support.

Conclusion

Is a good network security policy the end of all your problems? Not likely. But it's a start on a long and never-ending road. A static policy can be just as damaging as a bad policy. A good policy will be a dynamic living document and provide a good framework for the details that follow in standards, procedures, and guidelines. Once the policy is defined, the real work begins with implementation of that policy across various environments. As the saying goes, "The devil is in the details."

using trinux for security testing

by Dario Forte

Dario Forte is a security analyst for several Italian governmental offices and teaches information warfare/management and computer forensics at the university and post-university level.



<dario.forte@inwind.it>

Trinux is a light distribution of Linux, which shares a broader realm with other MiniUNIX such as tomsrtbt, LEM, and PicoBSD.

Trinux is booted from a single floppy, loads the rest of its modules from a FAT/Ext2 partition from other floppy disks or from an HTTP/FTP server, and runs completely in RAM. One of its most important features is that it includes a series of precompiled versions of security tools such as nmap, tcpdump, iptraf, and ntop. Furthermore, this distribution works by default with DHCP.

Trinux demands only modest hardware. The operating system will run on a recycled 486 with 32MB of RAM. This is sure to delight hoarders of old equipment. The kernel supports most network cards and is continually being updated.

Obtain GNU-licensed Trinux from <<http://www.trinux.org>>, which shows all the available FTP resources organized by geographic location.

Installation and Configuration

Since Trinux is a floppy-based distribution, the first thing to do is download the raw disk images from the FTP site and copy them onto the boot disk. This will take care of loading the kernel, mounting the first ramdisk, creating the additional ramdisks, configuring the network, and loading the rest of the packages.

The images can be inflated either with gunzip (UNIX) or Winzip. Below are some of the basic steps we used in our test, as recommended by the designers themselves:

1. Check the size of the images. You will need 1.4MB or 1,474,560-byte floppy disks, which will be completely occupied by the Trinux images. Since the files occupy all the space that is normally available on a floppy (a special program is used to transfer the images), it is a good idea to use clean floppies.
2. Linux users need to use the following command: `dd if=image-name of=/dev/fd0`. Naturally, you have to know which device to use. It should also be mentioned that the entire product was designed to be initially configurable and managed under Windows. Windows users will need a copy of rawrite, which can be obtained from <<ftp://ftp.trinux.org/pub/trinux/rawrite.exe>>. Install the utility in the same directory as the images to prevent wasting time on pathways.

NETWORK CONFIGURATION

Since Trinux is a network-centric operating system, it has to be used on the network to justify its existence, so you will have to configure the network card.

The boot floppy contains a script named `/init/netcfg` and is configured to use DHCP. As things currently stand, the project documentation indicates general compatibility with both UNIX and NT DHCPs. If you do not want to use the file, it can be moved into the directory `/conf/disabled` on the boot floppy or deleted. I would personally suggest the first option; the `dhcp` file may be missing for some reason. In this case, `netcfg` looks for two files in order to obtain information recently saved on the network: `/conf/eth0` and `/conf/network`. These are scripts that act on variables such as the IP address, subnet mask, default gateway, and so on. You should review the attached project documentation very carefully if you want to personalize the information contained in the files.

In general, remember that the Trinux boot floppies are MS-DOS and thus can also be configured by Windows. WordPad may be useful if you need to do some editing.

Useful notes:

- Unless modifications are made to the file `/conf/pkgconf`, Trinux loads the packages from one or more floppies. To operate otherwise, specify how to proceed. For example, if you want to interact with an HTTP/FTP server, create an empty file called `netload` in the directory `/conf` and insert the complete URL of the server from which you will be downloading.
- Likewise, loading the packages from a hard disk (more advisable) requires you to work with the `pkgsrc` file to indicate on which device Trinux can find the packages. In the case of a DOS/FAT partition, for example, you generally specify `/dev/hda1`. You must also determine the filesystem and the pathway to the files.

Tools Included in Trinux

Once installed and configured, Trinux is ready to use. One of the advantages for those who use Trinux, apart from being able to reuse the old computer, is the precompiled security tools.

Trinux elements are divided into categories. One part comprises packet sniffers, in particular, `tcpdump`, `ipgrab`, and `ngrep`. Each one has special features to let you get the most out of them. The network monitors in this case are `iptraf` and `ntop`, the latter certainly one of the most interesting currently in circulation. Netmapping and vulnerability-scanning tools such as `nmap`, `exscan`, `saint`, `queso`, `hping`, `firewalk`, and `cgichk` are also included. These are today's standard tools for the kit of any self-respecting security analyst.

Trinux also has firewalls and proxies, including the noted `ipchains`, `redir`, and `tinyproxy`.

Lastly, two tools are included from a chapter in the Tiger Team bible: `netcat` and `hunt`. These test eventual vulnerabilities to connection hijacking.

Special attention should be paid to the use of X Windows under Trinux. Given the types of tools included, something like this should not be necessary, partly because the tools (except for `ntop`, which, in spite of several recent security bugs, is very well respected by the technical community) do not use graphical interfaces in the true sense of the term. Practically speaking, it makes sense to use X only when you intend to deal with a plurality of X term windows contemporaneously. Moreover, at present, a mouse cannot be used; a workaround was adopted for using the keyboard in its place. You will also have to be content with only 16 colors.

The `ntop` program can be managed remotely from an HTTP console. However, as things currently stand, we do not know of any cryptosystem protection of the connection between `ntop` and its console.

Conclusions

I had my first exposure to Trinux a year ago along with two friends (currently Italian managers of two important American security companies). We had our little laugh about the future prospects of the project. Now, in spite of the fact that my two friends and colleagues see me as laboring under an illusion, Trinux has made it into the top ten free security tools worldwide. I have decided to launch an FTP mirror of `trinux.org` on my Web site. Matthew Franz, the mind behind the project, will be delighted to receive your comments and requests to participate in the initiative.

[See images on next page.]

Trinux has made it into the top-ten free security tools worldwide.

```

xterm
Port: 109 Open: Post Office Protocol 2 Service Running.
Data Returned:
+ POP2 localhost v4.46 server ready

Port: 110 Open: Post Office Protocol 3 Service Running.
Data Returned:
+OK POP3 localhost v6.50 server ready

Port: 111 Open: SunRPC Service Running.
Port: 113 Open: Authentication Service Running.
Port: 139 Open: NetBIOS Session Service Running.
Port: 143 Open: Interim Mail Access Protocol 2 Service Running.
Data Returned:
* OK localhost IMAPrev1 v11.241 server ready

Port: 513 Open: login Service Running.
Port: 514 Open: rmd Service Running.
Port: 635 Open: NFS Mount Service Running.

Scan Completed Successfully.
[root@localhost ~]#

```

```

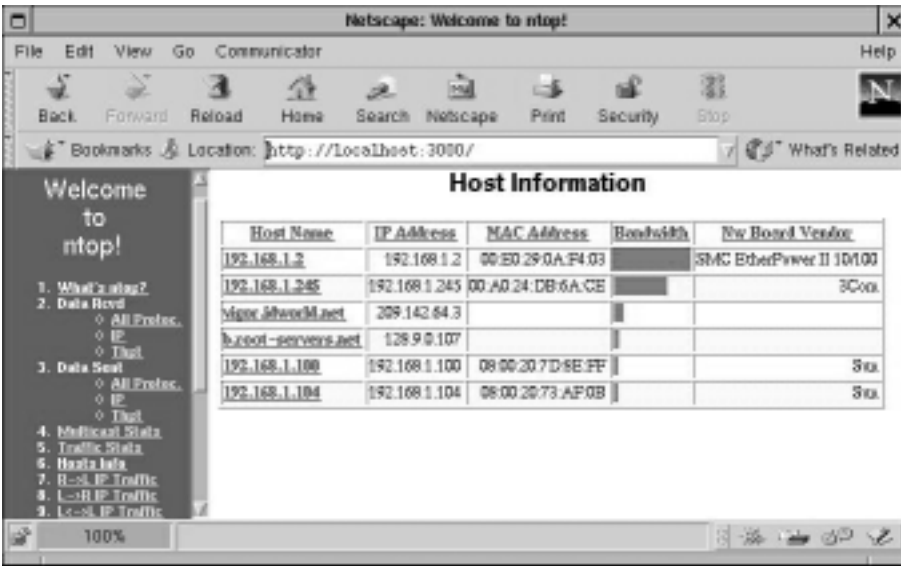
xterm
513 open tcp login
514 open tcp shell
515 open tcp printer
540 open tcp uucp

Interesting ports on (192.168.1.106):
Port      State Protocol Service
7         open  tcp     echo
9         open  tcp     discard
13        open  tcp     daytime
19        open  tcp     chargen
21        open  tcp     ftp
23        open  tcp     telnet
25        open  tcp     smtp
37        open  tcp     time
78        open  tcp     finger
111       open  tcp     sunrpc
512       open  tcp     rmd
513       open  tcp     login
514       open  tcp     shell
515       open  tcp     printer
540       open  tcp     uucp

```

nmap is Trinux's cutting edge. It is surely the most reliable portscanner, with numerous added features such as OS detection. It is also available under Windows NT, but does not enjoy the same stability.

exscan is an alternative portscanner to nmap that also captures login banners.



ntop can also be run via HTTP. If you use it off the Trinux machine there is no graphical interface.

```

xterm
ntop v.1.1a9 [i586-pc-linux-gnu] listening on eth0
10/12 Pkt: 706,0 Kb [IP: 704,1 Kb/Other: 1,8 Kb] Thrt: 24,5 Mbps/33,2 Mbps
Host      Act  -Recv-  Sent  TCP  UDP  ICMP
192.168.1.2      1  436,6 Kb  98,2 Kb  388,4 Kb  48,2 Kb  0
adric.genocidr2600.com  1  39,7 Kb  315,7 Kb  39,7 Kb  0  0
192.168.1.245    1  10,3 Kb  48,5 Kb  0  10,3 Kb  0
home.idcor1d.net  1  9,1 Kb  43,4 Kb  9,1 Kb  0  0
vigor.idcor1d.net  1  4,3 Kb  7,0 Kb  4,3 Kb  0  0
vanuz.erf.com    1  1,7 Kb  5,0 Kb  1,7 Kb  0  0
opensource.org  1  1,3 Kb  17,4 Kb  1,3 Kb  0  0
imagecarv.inqiz.com  1  904  0  904  0  0
www2.buou.com    1  78  8,3 Kb  0  78  0
E.root-SERVERS.NET  1  78  165  0  78  0
ns2.idcor1d.net  1  76  171  0  76  0
F.root-servers.net  1  68  0  0  68  0

```

a note on security disclosures

In recent months, a handful of outspoken security professionals have begun to openly challenge the philosophy of full disclosure. For years, most of us in the security community have held this philosophy as a basic tenet of vulnerability management. Software vendors have a notoriously bad track record in handling bugs and vulnerabilities in their products. Rather than practice due diligence and handle the incident swiftly and openly, vendors have not given these matters the attention they deserve. Open communication between security consultants or hackers who find these vulnerabilities and the vendors they report to is more of a myth than anything, despite what the vendors would like you to believe. Because of this, full disclosure has kicked in, causing security types to release all of the details to public forums. In some cases, they include working exploit code as proof of concept, to “gently” prod the vendor into dealing with the vulnerability in short order.

There are essentially three levels of disclosure with regard to security-vulnerability information seen today:

1. General information indicating a vulnerability in a specific software package or operating system. Often the information is extremely vague and not adequate for administrators to fix the problem themselves. Advisories from CERT and similar outfits fit in this category.
2. Full technical information, often in the form of an advisory. Technical details enable security personnel to understand the problem fully and often fix it without any additional information. No exploit code or “proof-of-concept” code is included. Security firms and consultants typically release in this fashion.
3. Working exploit code. This is sometimes accompanied by a technical advisory explaining the vulnerability in depth. Other times it may only include a few notes in comments at the top of the program.

The recent argument suggests that including exploit code while disclosing vulnerabilities has a downside that severely outweighs any upside. The perceived upside is that by including working exploit code, vendors are forced to respond to the issue quickly, lest thousands of their customers remain vulnerable to the new exploit. While this indeed puts extra pressure on the vendors to fix the problem, it potentially leaves thousands of systems exploitable afterward. While software patches are available, there is nothing forcing administrators to install them to negate the problem. At this point, full disclosure becomes a two-edged sword.

The downside to all of this is that these exploit scripts and utilities are available to anyone with a shred of computer know-how, who now has the capability of breaking into foreign systems. This in turn leads to the “script kiddie” phenomenon: large numbers of mostly unskilled wannabe hackers use the tools to break into (and often deface the Web pages of) systems around the Internet.

Much to the joy of some of these outspoken security professionals, there exists hard data to back their claims. For months, they have been making these claims with no real backing to support their arguments. As with any person making claims with no proof, it is often easier to discount their words as a bit fanatical, especially when their tone is elitist and condescending. Using data collected by Attrition (<<http://attrition.org>>) along with BugTraq archives at Security Focus (<<http://securityfocus.com>>), several

by Brian Martin

Brian Martin is a Senior Security Engineer with the DSIC Network Security-Group. His team provides penetration assessment and audit for the commercial and government sectors.

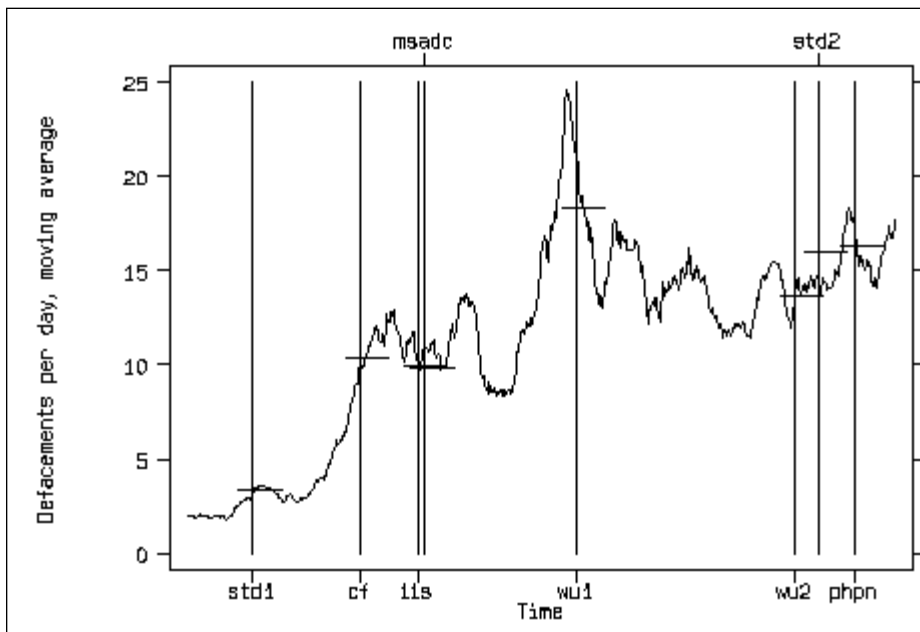


<bmartin@attrition.org>

cases have clearly emerged that demonstrate the cause and effect between exploit-code release and Web-page defacement.

Before examining this data, several things should be considered. This is not the silver bullet killing any and all doubt you may have had in the outspokens. Rather, this is data that proves one piece of the puzzle. As with all things, this is not a black-and-white issue. Other issues must be considered in addition to the ideas presented above.

Several cases stand out in the past two years that demonstrate the downside to releasing exploit code. Listed below are eight vulnerabilities that allow an intruder to gain some form of elevated privileges remotely. Included with each is the approximate date exploit code was made public, along with other thoughts or comments.



Vulnerabilities, November 1, 1998 - September 30, 2000

Note: Horizontal lines are the average defacements per day 14 days before and 28 days after vulnerability became "public" knowledge.

"std1" – automountd/statd remote buffer overflow (UNIX) Jan 4, 1999
 <<http://www.securityfocus.com/archive/1/11788>>. Shortly after public disclosure, there was a small spike in defacements per day. In the following months, an incredible growth began.

"cf" – Cold Fusion l0pht advisory w/exploit code (NT) Apr 20, 1999
 <<http://www.securityfocus.com/archive/1/13377>>. Other problems in the Cold Fusion package came to light in Phrack 54 (Dec 25, 1998), but did not include detailed exploit information. Based on the graph, it seems the release of the CF exploit resulted in more defacements per day.

"iis" – IIS Hack eEye advisory (NT) Jun 16, 1999
 <<http://www.eeye.com/html/Advisories/AD19990608-3.html>> and <<http://www.securityfocus.com/archive/1/15448>>.

"msadc" – RDS/MSADC RFP advisory (NT) Jun 23, 1999
 <<http://www.wiretrip.net/rfp/p/doc.asp?id=1&iface=2>>. The combination of IIS Hack and the MSADC

exploit being released at approximately the same time led to two small spikes. Because of difficulty in getting the exploit code to work early on, it is believed that the incredible spike in the following months was more indicative of the exploits being public. During this time, a large percentage of defacements mirrored by Attrition appeared to be NT-based and mostly a result of the MSADC vulnerability.

"wu1" - wuftp 2.5 remote buffer overflow (UNIX) Nov 20, 1999
 <<http://www.securityfocus.com/archive/1/35828>>. While the average number of defacements per day dropped steadily shortly before and after its release, there was another noticeable spike shortly afterward. Once again, it is believed that the delay was caused by initial problems in using the first versions of the exploit code. In the weeks after its release, more versions of the exploit came out, increasing the chances of successful exploitation on a remote host.

"wu2" – wuftp 2.6* remote buffer overflow (UNIX) Jun 23, 2000
 <<http://www.securityfocus.com/archive/1/66367>>. As seen before, a small increase can be

seen before and after the release of the exploit code. Running into the approximate release of “std2,” the upward growth became even more noticeable.

“std2” – statd remote buffer overflow (UNIX) Jul 16, 2000

<<http://www.securityfocus.com/archive/1/70306>>.

“phpn” - PHP-Nuke news site administration Aug 21, 2000

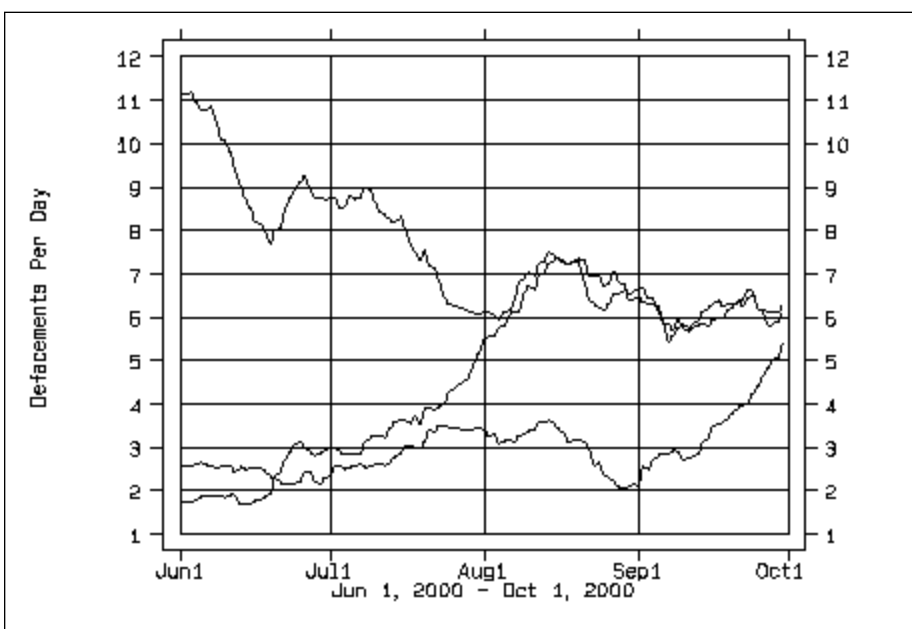
<<http://packetstorm.securify.com/0008-exploits/PHP-Nuke.c>>. Once again, a noticeable spike shortly after disclosure of the exploit information. During this time, a large percentage of defacements reported to Attrition were a result of this exploit. Because the attackers could post custom messages to a news application and not replace the entire page’s content, it was rather easy to identify which defacements were a direct result of this vulnerability.

While these eight examples are fairly clear, it should be noted that with the disclosure of any remote exploit code, defacements tend to increase shortly afterward. Depending on the operating systems affected, ease of use of the exploit, and availability of vulnerable machines, the numbers do not always shift so dramatically. Working with the Attrition mirror on a daily basis makes one more aware of this trend.

A key part of the definition of “script kiddie” is the lack of technical skill s/he possesses. It is widely believed that most script kiddies use Windows machines and favor exploitation of Windows NT servers. In the cases where a UNIX-based exploit is simple enough to use (easy to compile, simple command-line arguments, etc.), script kiddies will shift from exploiting Windows machines and begin to attack UNIX systems. The best example of this can be seen in the recent “wu2” (wuftpd 2.6) and “std2” (statd) vulnerabilities. Not only did significant spikes occur shortly after public disclosure of exploit code, but a radical shift in the overall number of Windows and UNIX operating systems being defaced occurred.

Despite a steady increase in Windows NT defacements for the last year, NT systems were defaced less often shortly after the two UNIX exploits were released. In keeping with this, Linux (exploit code for wu2/std2 was written to exploit Linux primarily) defacements climbed dramatically. Defacement groups that had previously been dominant on Windows platforms suddenly began to deface more and more Linux machines.

While the data points to a conclusion that publicizing exploit scripts is harmful to the Internet community, that may not necessarily be the case. Because Web-page defacement is a public event and a strong motivation of many script kiddies, it provides a method to extract data to show the trends and statistics above. However, one must consider the sequence of events and results of exploits being created but *not* being posted to a public forum.



Operating Systems, June 1, 2000 – October 1, 2000
(Top Line – Windows NT, Middle Line – Linux, Bottom Line – All other OSs)

No fewer than 20 defacements occurred in the process of writing this article.

Unpublished exploit scripts are like currency in the hacker subculture. The power of a single remote exploit that is unknown to vendors enables a single person to potentially break into thousands of machines, often with no recognizable trace of how it was done. Many intrusion-detection systems will not recognize the fingerprints of these new exploits. The hackers who hold these scripts typically do not deface Web pages or perform any action that would draw undue attention to themselves. To do so would bring more attention to their actions and create a better chance that someone would figure out how they are compromising machines and what the new vulnerability is.

In some cases, these exploits circulate in the underground for up to a year before being made public. If even a dozen hackers have such an exploit while actively using it over a one year period, the damage becomes negligible compared to a few dozen Web defacements that occur as a result of an exploit being made public. While it is unfortunate to see a site hacked as a result of the public disclosure of a vulnerability, it really becomes one of the necessary evils in doing so. One has to balance the sites being hacked in one hand versus a vendor patch that will allow thousands of sites to be protected against the exploit.

As I wrote this article, I had constant reminders of everything covered above. No fewer than 20 defacements occurred in the process of writing this. No doubt some occurred as a result of exploit scripts being made public, fueling the script kiddies. Equally so, some of these probably occurred from classical attacks such as password guessing, sniffers, or nonpublic exploits that haven't crossed BugTraq yet. Is releasing exploit code as evil as some are now saying? I think that is answered by which side of the sword you'd rather be cut with.

[Editor's Note: My home systems were penetrated within two hours of the Kerberos exploit being published. I think it was the first BSDI had heard of it. Even having the fix within four hours wasn't enough to prevent the exploit. RK]

mail message metering

The scourge of spam has existed since the commercialization of the Internet began in the early 1990s. Unfortunately, spam is here to stay irrespective of any legal, procedural, and technical measures used to contain it. ZipLink, as a wholesale provider of Internet access, is in a unique position to do its part in stopping spam from emanating from its network, saving other network operators and end users time, money, and inconvenience.

The March 6, 2000 edition of *Interactive Week* contained an article outlining the bankruptcy of AGIS, a wholesale provider of Internet access and harbinger of spammers. In the article author Max Smetannikov states that Lawlor, president of AGIS, “opened AGIS to unsolicited commercial e-mailers and only relented after a walkout of key technical staff and a crippling hack attack in 1997.” The issue of spam is critical for organizations like ZipLink. If we don’t try to eliminate spam, our transit providers and peers will shut us down, causing our business to cease operating, as almost happened to AGIS.

The problem of unsolicited commercial email (UCE) from a wholesale provider’s perspective is two-part:

- how to reroute all mail traffic to a single point (mail relay) without causing undue stress on our wholesale customers and end subscribers, and
- how to identify a bulk mailer in progress.

The first problem is solved by redirecting SMTP (outbound mail) sessions to centralized mail relays at appropriate points throughout our network. The second issue depends on how one defines a spammer. ZipLink believes that a user who exceeds a certain message count per unit of time can be accurately defined as a spammer in progress.

Our goals for a technical solution for the problem of spam include:

- blocking at least 50% of outbound spam
- creating little or no impact on customer ISP and end user
- having the ability to exclude certain domains and users
- providing a solution that is configurable and scalable
- minimizing the impact on existing infrastructure
- utilizing free software where possible.

Of course, blocking spam is the most important goal. While in-production numbers don’t yet exist, we believe 80% of all outbound spam can effectively be blocked with our unique method. Limiting the impact on the customer ISP and end subscriber is important. As a result, our solution includes the use of layer-four switches to automatically redirect SMTP connections to centralized mail relays without intervention by ZipLink, the customer ISP, or the end user.

Since many of our ISP customers do a very good job at blocking UCE on their own and have explicitly asked to be excluded from our anti-spam solution, any anti-spam filtering method put in place must include an opt-out on a customer-by-customer (ISP) basis. Also, some end subscribers have legitimate reasons for sending large amounts of email that may appear to an observer as UCE. These bulk mailers must be able to send their legitimate mail on an ongoing basis without continuous intervention.

Of course, any solution cannot render our existing infrastructure unusable, so it must have little impact on our RADIUS (Remote Authentication Dial In User Services,

by Robert Haskins

After a 10 year career as a UNIX system administrator, Robert Haskins is currently Director of Information Technology for Ziplink, Inc. He still occasionally dabbles on UNIX systems much to the chagrin of his staff.



<rhaskins@ziplink.net>

defined by RFC2138 and RFC2139) servers. Lab testing indicates an impact of approximately 5% on our RADIUS servers, which is well within our goal. Finally, it is important to leverage existing solutions in order to reduce development time and cost, if at all possible.

Existing Solutions for Spam

The existing solutions for curtailing spam fall into four general categories:

- services, such as Brightmail and MAPS' RBL
- client-based solutions, such as "POP before SMTP" and Hash Cash
- server-based solutions, such as Blackmail and SpamShield
- RAS filters.

Service-based solutions are aimed squarely at the end user or ISP providing end-user services, which ZipLink does not provide as a wholesaler of Internet access. Client-based solutions are too expensive in terms of end-user support costs and don't meet our requirement for minimal impact on end users. Existing server-based solutions do not have the required flexibility and are aimed again at the receipt of spam, not at sources of spam such as wholesale access providers.

Many wholesale providers today, including ZipLink, utilize RAS filters to block UCE. While such filters work, they have at least three major limitations:

- Filters are not manageable with 500 ISPs (since at least one unique filter is required per ISP).
- Filters significantly decrease dial-up performance of the RAS equipment and therefore can impact the customer's online experience significantly.
- RAS equipment has finite memory available for filters, which limits the number of filters that can be applied across the network.

These shortcomings severely limit the usefulness of filters on RAS equipment as a method to block spam. However, they do serve a purpose and will likely be used for certain customers (free/subsidized ISPs, for example) in the foreseeable future.

Mail Message Metering

Under ZipLink's patent-pending method of blocking spam, UCE is defined as large volumes of messages sent over a short period of time. Spammers don't normally send a message per hour; they usually try to send as many messages as they possibly can over the shortest period of time.

Figure 1 shows a block diagram of our solution in its entirety. The basic premise of the system is that a particular IP address may originate only X number of messages during Y amount of time. A message is defined as a to:, cc:, or bcc: recipient in the header of the message. For example, if the user had a to: line with three addresses and a cc: line with three addresses, a total of six messages would be charged against the user's quota. In order to allow exceptions to the defined maximums, we track the user assigned to the IP address via RADIUS accounting records. RADIUS records

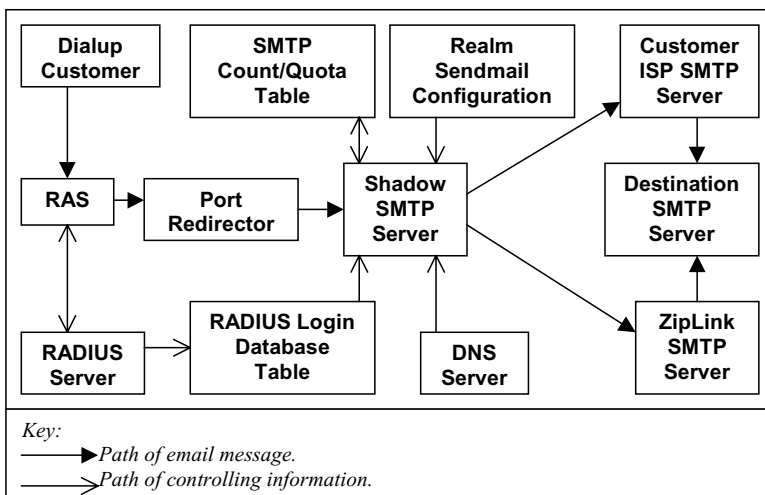


Figure 1

are placed into an Oracle database as the same record is concurrently entered into the usual RADIUS accounting file for redundancy purposes.

To illustrate exactly how the process works, it is useful to track the path of a mail message through the system. As an end subscriber initiates an outbound mail connection, a layer-four switch redirects the session to a shadow SMTP server. That mail server performs a quota lookup in the Oracle database and determines whether or not this user has exceeded its user, domain, global, or compiled in (default) limit. If the user's quota has not been exceeded, the message counts in the Oracle database for that user and domain are updated and the message is allowed to pass. If the user has exceeded its limits, the customer receives a "450 Mail quota exceeded for %U" message.

The layer-four switch contains configuration logic that can except certain domains from being forwarded to the quota-checking mail relays. As a result, ISPs/domains who do a good job and don't cause spamming problems can easily be excepted from the system. Of course, the ISP can easily be added back should it stop handling UCE appropriately.

Getting RADIUS Records into the Oracle Database

It is worth noting here that one could easily forego the saving of RADIUS data (or other IP address-to-user mapping) into an Oracle database if one didn't have such data. The result of doing this would be an inability to allow certain end users or domains to exceed default values. In addition, the RADIUS data could easily be replaced with DHCP or static IP address data. The reason RADIUS data is used is that our RAS equipment utilizes RADIUS for authentication, authorization, and accounting, as do most (if not all) ISPs.

Figure 2 outlines how data enters the Oracle database. Each RADIUS server runs a program called "radius2db," which is a series of Oracle function calls that forwards RADIUS data into Oracle. This program is approximately 800 lines of C code and was written entirely in-house by Dale Nielsen. A MySQL version of the code exists, and should ZipLink decide to open-source this solution, this version will likely be released.

In our preliminary testing, we have shown that the impact on our RADIUS servers meets our requirements for limited impact. In fact, the process related to this activity takes approximately 5% on both a dual-processor 300 Mhz Sun Ultra 2 with 768 MB of memory and a 300 Mhz Sun Ultra 5 with 512 MB of memory. The impact on our Oracle database will be tested once we go into limited production testing. Since Oracle is relatively easy to scale, impact on it is less important. It is expected that the existing Oracle infrastructure (a Sun E450 with a single processor and 1 GB memory) will easily be able to handle 100,000 ports of traffic, which is approximately double the size of our existing network.

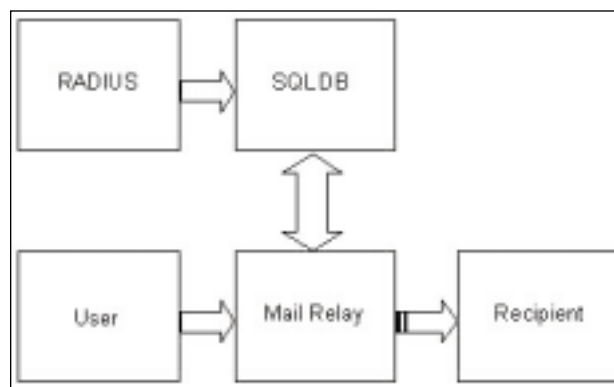


Figure 2

Quota Checking Mail Relays

When we evaluated solutions for our quota checking mail relays, we closely examined two firewall proxy SMTP solutions: the Trusted Information Systems (TIS) Firewall Toolkit SMTPD and the Obtuse Systems Corporation Juniper Firewall Toolkit SMTPD. While very similar in functionality and features, we picked SMTPD from the Juniper Firewall Toolkit for the following reasons:

- The license is “BSD style” and therefore less encumbered than the TIS toolkit.
- SMTPD runs as a daemon, which we feel is more reliable than running out of inetd as SMAPD does.

SMTPD performs the following functions on each mail message:

- verifies that source IP address is within the ZipLink range
- checks mail quotas for that user
- forwards message to SMTPFWD.

SMTPFWD, part of the Juniper Firewall Toolkit, simply utilizes sendmail to forward the message to the ISP customer’s mail relay. Although each ISP customer’s mail relay needs to be entered into the sendmail configuration, no changes were made to the sendmail source code. The failure mode of the quota-checking mail relays is to allow the message to pass if it cannot reach the Oracle database.

Tunable Parameters

In order to enable per-domain and per-user limits, quotas must be kept in the Oracle database. The quotas are checked in the following order:

1. number of messages per time interval for user@realm
2. number of messages per time interval for @realm
3. number of messages per time interval, global
4. the compiled in default is 10 messages/10 minutes, max 100 messages/24 hours.

If the first limit (number of messages per time interval for user@realm) doesn’t exist, then the check drops to the next limit. For each level, there are two sets of checks done. For example, there might be a limit of 100 messages per 60 minutes and 1,000 messages per 24 hours for user@isp.net. If either one of the limits is exceeded, then the message is not forwarded. If neither quota has been met, then the message counts for that user are updated to include the message, and the message is forwarded on to the customer ISPs mail relay for final delivery to the end recipient.

For the majority of subscribers, the ISP can simply set a domain limit and that is all. For legitimate bulk mailers (such as opt-in mailing lists) the ISP can set that particular customer’s limits higher to an appropriate threshold. The customer, the customer’s ZipLink account manager, or the ZipLink NOC can make changes to limits via a simple Web-browser interface.

Limitations

As with any complex system, nothing is perfect. Having a solution like this for the problem of UCE, it might be tempting to eliminate anti-UCE language in the Acceptable Use Policy (AUP). However, this system does still require an AUP. In fact, it would be a good idea if all ISPs placed message-count limits directly in their AUP. This might go a long way to reduce UCE.

Another problem is that the limits set within the system can be circumvented. This might be the case if the ISP customer is sympathetic to its customers who generate UCE intentionally for profit. The easy fix to this is to not allow such ISPs to change their customers’ limits.

One possible problem with this system is that end customers cannot relay mail through arbitrary mail servers. An example of this use would be an employee of a company

working from a home office who would like to relay mail through his company's mail server. We're not sure if this poses a problem or not, but if it does, steps can be taken to enable certain customers this ability. This may require modifications to the sendmail code, but further analysis is required in order to complete a design.

One of the toughest problems is the fact that this design requires the use of layer-four switches to redirect SMTP sessions to the quota-checking mail relays. This is not a trivial problem for providers the size of ZipLink whose networks have not been engineered for this functionality. This will require the reengineering of every ZipLink point of presence to ensure that all traffic goes through a central point.

The final issue that we see with this system is that pro-First Amendment organizations such as the Center for Democracy and Technology and the Electronic Frontier Foundation may very well argue that we are impeding our customer's First Amendment rights. As stated previously, the problem of UCE is a matter of business continuity for most ISPs and must be addressed.

REFERENCES

Trusted Information Firewall Toolkit:
<<http://www.tis.com/research/software>>

Blackmail:
<<http://bitgate.com/spam>>

Obtuse Systems Corporation
Juniper/smtpd Firewall Toolkit:
<<http://www.obtuse.com/smtpd.html>>

Brightlight Technologies Brightmail:
<<http://www.brightlight.com/isp/spam>>

Mail Abuse Prevention System (home of
RBL/TSI/DUL/RSS):
<<http://mail-abuse.org>>

Hash cash:
<<http://www.cypherspace.org/~adam/hashcash/>>

Spamcop:
<<http://spamcop.net>>

Spamshield:
<<http://spamshield.conti.nu/>>

technical interviews

by **Strata Rose Chalup**

Founder, VirtualNet Consulting; Director of Network Operations, KnowNow Inc., Strata Rose Chalup got her first sysadmin job in 1983 and never looked back.



<strata@virtual.net>

I was inspired to write this article when I read a piece about technical interviewing that seemed, frankly, to have its head up its /dev/null. An online pointer to said article was forwarded to a social/technical mailing list of which I am a member, and a couple of folks said, “Do you think this is reasonable?” to the assembled multitudes. As a 100% card-carrying lurker, I tend not to respond to things unless they hit one of my hot buttons. One well-established hot button is any of a large class of articles on hiring and interviewing that I have seen in the trade press lately.

These articles are geared toward new and established managers, and purport to deliver “the essentials” of the hiring process. The vast majority of them seem to make several assumptions that are insulting to most of the participants, including the company doing the hiring. Common themes include:

- Most candidates are bad candidates.
- Candidates are hungry for a job with bigcompany.com, the bigger, the better. Your name/brand alone is sufficient to woo them.
- Candidates who have multiple interviews with multiple firms at the same time are not playing the game fairly, or are inexperienced. Candidates should go through the process first with bigcompany, then with nextcompany, etc.
- The interviewing team is there mostly to see if the candidate knows his or her stuff technically. You can assemble one ad hoc from any combination of potential coworkers.
- The interview process exists primarily to weed out bad candidates.
- The candidates will be positively impressed by how ruthlessly you pursue the weeding process.
- After you weed out the bad ones, just pick the best candidate from the remaining pack.

Anyone who has hired sysadmins will realize how profoundly bleak and inaccurate a picture these worldviews present. Let’s see if we can combine some less degrading worldviews with some practical advice. If we’re lucky, the combination will help those of you looking for new hires to find and attract good ones. If not, we’ll get a whole pile of flame mail. I already have a button that says, “Oh no, not another learning experience.” Yee-ha-ha! Full speed ahead, and `ctrl()` those torpedoes.

Not a Unique Problem

You can’t get what you want until you know what it is, right? We are usually looking for a rare combination of independence and diligence, ability to groundbreaking but also to follow procedures. In some cases, depending on the position, what a candidate already knows is much less important than how the candidate thinks and acquires new knowledge. Of course, they’d better already know “enough,” or they’ll spend too much time coming up to speed.

As a profession, we tend to feel that we are fairly unique, but in fact the medical profession has some very similar needs. An acquaintance in that field described the process of hiring clinical staff, saying, “What I want are staff that are resourceful and able to problem solve to some degree on their own. We definitely set guidelines as to what they are and aren’t able to advise patients on, but a little independent thinking on their part allows us to work without constant interruption. They have to be able to triage the urgent problems and do so without panicking (at least not in front of the patient!).” Sound familiar?

BABY-SIZE HOLES IN /DEV/HIRING-FILTERS/BATHWATER

We all like to joke along the lines of “Where are you working? – Hmm, what month is it?” – but few people enjoy the disruption that comes with settling into a new position frequently. Our own SAGE salary survey shows that other factors besides money are often the prime factors in job changes. Good sysadmins are a scarce resource, and they know it. They usually view shopping for a new job as a pain, rather than a necessary once-a-year salary increase, and will put a great deal of thought into picking the position that seems to emphasize fun and learning.

Too many participants regard the fundamental purpose of technical interviews as a winnowing or weeding-out process. This kind of attitude leads them to structure the entire process as a series of ever-higher hoops that the candidates have to jump through before they are judged good enough for a second round, let alone an offer. Unfortunately, this kind of tough-guy interviewing often backfires, giving the impression that your workplace is arrogant and uninteresting. We need to consider two fundamentals, not just one. “Screen each candidate rigorously” is the obvious first. The oft-neglected second is, “While doing this, how do we simultaneously convince them that we have a good work environment?”

I don’t for a moment suggest that one should relax one’s standards for technical contribution. The truth is that it is difficult to switch horses in midstream from “we’re not sure you’re good enough to work here” to “we’re such a great place and that’s why you should work here!” At least, it’s difficult to do it and not sound completely fake – I’ve certainly seen it done many times, just rarely seen it done well. The interview process should be designed to convince every candidate, even the ones who are hopeless, that you have a great team and a great workplace.

IT’S THE MINDSET

If you view this as “wasting effort,” you are already part of the problem. Even the worst candidate has friends who are more talented – perhaps especially the worst candidate. People grow and change. Here in Silicon Valley, a year can make a huge difference in someone’s skills and maturity level. When Joe Clueless turns into Joe Admin, you would assuredly rather have him resubmitting his resume to your group than telling all his friends, “Oh, I interviewed there a year ago and they were a bunch of fascist losers; don’t bother!” Even better, when Joe Clueless goes to one of the many technical group meetings or conferences, he might tell Jane SuperAdmin, “Hey, I interviewed at FooGroup the other day. They were looking for someone a lot more like you than like me, and they looked like a really fun place to be!”

Please note that I said “participants” a few paragraphs above. Usually the candidates, as well as the interview team, are often indulging in this kind of zero-sum thinking. To be sure, only one person will get the job – this job. What about the next job, for a more junior or more senior candidate? What about the job at the company where an interviewer’s spouse, friend, or colleague works? Or the job a year from now at the candidate’s employer, where the candidate needs someone like you on his or her team? One of the most valuable pieces of “technical” advice I’ve ever received was given to me by my first real IT manager, Alan Wu at MIT-EECS. He told me, “It’s a small world, and nobody ever really goes away. That means that nobody is disposable, because you’ll be running into them again and again.”

Think of the interview as merely the first time that you will encounter a particular candidate. If you are both good, the chances are that you’ll see each other again. It may even be on a different side of the interview desk.

Too many participants regard the fundamental purpose of technical interviews as a winnowing or weeding-out process.

Brief your human resources people about what characteristics and experience are important to you.

The Interview Life cycle

CREATE A JOB DESCRIPTION

Include any hard and fast requirements right in the body of the description, like Oracle experience, sendmail experience, etc. Describe things in the most specific terms possible – they will probably make you edit it down, but at least you have it clear in your head now!

Depending on your group style and responsibilities, your team may help you come up with the wish list. Be sure you distinguish between “must have” and “would be nice.”

If you are a believer in certification, and have chosen to require it for a candidate to be considered, be sure to specify the type and how to recognize it for HR, e.g., “Linux certification” versus “Red Hat certified” versus “must have Red Hat XYZ certification, would like Red Hat ABC certification as well.” Bottom line: anything that is very important to you will need to be specified in ways that a nontechnical recruiter mining a bad database engine can pull out for you, or you will miss good candidates!

CREATE A REQUISITION OR BUDGET ITEM FOR THE POSITION

This one is really crucial, especially if you know potential candidates. Don’t go getting your friends’ hopes up until you do! As part of this process, you will have identified who is the hiring manager and who is the reporting manager. Yes, Virginia, due to extreme trust or extreme department politics, they may not be the same person. You, or someone who can represent you, must have a budget for the candidate. This is generally the toughest part of the process, as the necessary paperwork and approvals can literally take weeks.

You may think, “Shouldn’t I get the budget first, then do the description?” Perhaps, but we are not all that fortunate. In many organizations, it is necessary to create a job description first, including the duties to be performed, to demonstrate that there is a need for the position. You may end up crafting a two-part document, in which the first part is the traditional job description and the second part is the job justification, including a detailed description of what currently neglected tasks will be performed by the new hire.

MAKE YOUR REQUIREMENTS KNOWN TO HR

Brief your human resources people about what characteristics and experience are important to you. Unless your organization is atypical, you will need to make sure they are not using some kind of lame keyword-driven resume-search service, and you will probably need to show them Internet resources like the SAGE Jobs Board, Unix Guru Universe, Monster.Com, Dice.Com, and so on. When you talk to HR, make it a two-way conversation.

You need to be sure of the requirements for introducing a candidate into the process, in the event that you or a team member will be referring someone, or if you encounter a potential candidate’s resume online. In some firms, you cannot just snag a resume off the Net and call the person. There may be a requirement to have HR perform some initial screening. Be sure that you include salary information for the position in the material you give to HR.

DO THE “TEAM HUDDLE”

Talk to your existing team about the position, if you haven’t already. It can be quite unsettling for a group to suddenly be told “here is your new coworker,” as is the custom

in many organizations. A good manager will make sure that everyone knows why another pair of hands is needed, and what responsibilities will be transferred to the new position. He or she will also work with employees whose job duties are directly affected by the new position.

In many cases, the position will have been created in response to overload on the part of existing employees. This does not absolve a manager from taking time with each of them to make sure that the new position is seen in a positive light, and to do a sanity check on how the work will be apportioned. I have heard a number of tales of people who were wearing several job hats they detested and one or two that they liked well enough to keep them at the job. Figuring prominently in these tales is a well-meaning but clueless manager who transfers their favorite hat to a new hire to “take off some of the pressure.”

THE NITTY-GRITTY PROCESS

1. Line up a first-round interview team.
2. Identify first-round questions and issues.
3. Establish a feedback form (if HR does not have one already) for evaluating resumes.
4. Farm out any resumes to the interview team, with a hard-and-fast deadline for returning feedback.
5. Identify candidates to interview.
6. Schedule interview time slots with your team.
7. Pass on the available time-slot schedule and the candidate list to HR and let them line up the actual interviews.
8. Do the first-round interviews.
9. Get feedback from the interviewers.
10. Choose second-round interview team.
11. Identify second-round questions and issues.
12. Do second rounds.
13. Get more feedback from the interviewers.
14. Identify candidates who will be eligible for offers.

“IS THAT YOUR FINAL, FINAL ANSWER?”

No one wants to be disappointed, but it is a good idea to identify a second (and perhaps even a third) candidate from among the qualifying candidates during your final feedback round. Attempting to do so later will be much less efficient, since people’s memories of the interview will have faded. It also can clarify whether you should start lining up resumes in case your first-choice candidate does not accept – the feedback may be that none of the other second-round candidates is quite enough of a fit if the first-choice one refuses.

Do not extend even a verbal offer until you have signoff from HR and/or your management chain! Common sense, but be sure to apply it. Your written offer should have a specific time limit on it, anywhere from a week to several weeks, depending on your situation. There are often good reasons why a candidate cannot make a choice immediately. Nevertheless, your organization needs to be protected from a situation in which all the other qualifying candidates move on because your “best fit” takes far too long to turn you down.

Be sure that you’ve gotten all the requisite background information from HR before doing your final feedback round. It’s incredibly discouraging to get through the process

Do not extend even a verbal offer until you have signoff from HR and/or your management chain!

If you don't hear back from the candidate by the agreed-upon date, don't hesitate to call.

and find that your best candidate is ineligible because another department scored your last H-1 visa sponsorship for this year.

THE END OF THE RAINBOW!

Really good candidates may have offers on the table already, so it's important for them to get yours as soon as possible. It's a good idea to make sure that a candidate gets an offer on a Friday rather than a Monday, even if you have to push HR to get it done. He or she will probably want to talk to their spouse, friends, and colleagues about their prospects over the weekend.

If you are the hiring manager, you should extend the offer personally by telephone (subject to HR rules in your organization). Be cordial and let the candidate know up front that you are not looking for an answer on the spot – you are merely delighted to be able to offer him or her the position. Indicate that you'd like to discuss it further in a day or so, asking if there is a specific date that they'd like you to call them. If they name a date too far in the future, negotiate! If the salary and/or benefits are flexible, also stress that you are available to discuss the offer in more detail if they have questions or concerns about it.

KEEP YOUR FINGERS CROSSED!

If you don't hear back from the candidate by the agreed-upon date, don't hesitate to call. If he or she does not return your calls, there's a good chance that you will need to contact other candidates. Be certain that you do not extend an offer to another candidate until the expiration of your first offer! If both candidates accept, you are in a sticky legal situation.

Hints, Tips, and Specifics

USE A WELL-ORGANIZED PROCESS

For yourself, make up a list of questions beforehand and use the same list for every candidate you are interviewing for this particular position. It will give you a much better set of points of comparison than if you "wing it" with each candidate.

Insist that other interviewers do a question list as well, and share them with each other. Don't wait until the feedback meeting to find out that several folks asked questions about the same area of expertise, and no one asked about other issues, or that you all asked pretty much the same questions.

Bring your notes to the feedback meeting. A no-brainer, but you'd be surprised how many people come empty-handed and then can't remember later if Candidate A or Candidate B seemed to know more about some subject. Hint: take your notes on the cover page (schedule page) you got with the resume, and keep them stapled together.

Line up time slots for consecutive interviewing with your team before the HR person calls the first candidate! People are hard to reach sometimes – your people and the prospective candidates. You don't want to lose someone good just because one of your key teammates is on vacation or is swamped and can't meet with the candidate. Weekly group meetings are a great place for choosing time slots, since conflicts can be worked out then and there. Your people do bring their calendars, day planners, or PDAs to your staff meeting, right?

If you are lucky enough to have a reliable shared-calendar system that your team actually uses, you can schedule time slots that way. It's poor form not to let folks know the days you are considering in advance, though – not everyone puts their dentist appoint-

ments on the work calendar. Be sure to allot half again as much time as you think you will need, to allow for variations in the times that candidates are available to interview.

Decide in advance what sorts of questions will be first round and what sorts will be second round. This helps keep the process cleaner, and also allows you to tailor the second-round questions as a group, based on the qualifying candidates.

If you're recommending someone, be honest with yourself – are they really qualified, or would you just like to be working with them (again)?

Some Basics

The candidate should be able to handle the technical questions presented in the first round. He or she should have a firm grasp of whatever basics you have decided are essential for this position.

Make sure candidates can draw and understand technical diagrams! They don't have to use Visio, but make sure they can look at something on a whiteboard and understand it. Be prepared to discover that the diagram equivalent of linguistic drift has occurred within your group, and don't expect the candidate to know that a flat box always means a router and a tall box always means a server. Just the basics.

If you're looking for a heavy scripter, at least ask the candidate to design a program right there in the office. Do it in pseudo-code, don't worry about syntax, but just do it.

ASK NOT WHAT YOUR SYSADMIN CAN DO FOR YOU . . .

Make it clear that you are interested in the candidate's total contribution to the team, both technical and team fit/social. Yes, you want the skills, but let's face it, most people who are truly competent already can write their own tickets. It's up to you to show them why they want to work with you and your fellow interviewers. I'm not talking bribes or happy-hour Fridays, I'm talking basic commitments to career growth.

Who goes to conferences? Who gets to take continuing education classes? Is there a book budget, formal or informal, with the position, and where do the books end up eventually? If the hours are routinely long, is there comp time? Finally, last but far from least, there's general experience – it should be pretty clear in the interview process that you and the candidate will *both* be able to learn a lot from working together. If not, what's the point?

TEAM FIT: MORE THAN JUST HAIR COLOR AND NAPSTER SETTINGS

The title of this section notwithstanding, it's worth mentioning that common sense, as well as a number of state and federal laws, should prevent you from judging people solely by their appearances. 'Nuff said!

Line up meetings before the interviews, meet and talk about exactly what you are looking for as a group – you, your teammates, and your manager, or you and the team if you're the manager. Decide what is more important, existing knowledge or easy learning, and decide which skillsets are crucial. Make explicit tradeoffs, since you can't have everything. Talk about what constitutes team fit for your workplace.

During the interview, if you are the manager, ask candidates what kind of status and reporting practices they have used in the past, and about their preferences. If your group is a “weekly status meeting, plus drop a note in email to the group when you do something big” kind of place, then you are not going to be a great fit for someone used to getting a set of specs and coding in the dark for a month, or vice versa. Let them ask you the same – seasoned candidates usually will!

Make it clear that you are interested in the candidate's total contribution to the team, both technical and team fit/social.

When interviewing someone, make sure you ask a combination of detail questions and process questions.

Be up front but not discouraging about any limitations of the job – on-call pager, tight deadline coming up, etc. People nowadays will walk unashamedly after a few weeks of a bad job fit, so you want to prevent that if you can. Things like whether the position is included in the on-call rotation, and how often, should have been defined at the very beginning.

Don't scare folks away, but be up-front about it. Make it clear that what you are talking about is the truth, *not* the tip of some evil iceberg. If you can't do this well, have the hiring manager do it, since he or she is usually expected to be "the heavy" in the interview process anyway. Yes, I said the same thing near the beginning of this article. It's worth repeating!

NOT "JUST THE FACTS, MA'AM"

When interviewing someone, make sure you ask a combination of detail questions and process questions. For some jobs, one is more important than the other, but generally you want both. Whichever they give you, accept it politely and encouragingly, and promptly ask for the other one.

Example Question: "I sit you in front of a Web browser and when you try to load a page, you are getting a "host not found" error. Tell me the kinds of things you'd look for to find out why that was happening."

Sample process answer: I'd check to see if the machine was set up to resolve hostnames properly, or if its network connection was down.

Sample detail answer: I'd check the `/etc/resolv.conf` file to see if it was present. If it existed, I'd use `nslookup` to try some name resolution by hand.

The Interview, the Actual Interview

This is no time to neglect the basics of public speaking. As they say, "First I'll tell you what I'm going to tell you, then I'll actually tell it to you, then I'll tell you what I told you, and then I'll take questions." This is a good way to handle the interview process.

WHO

Tell the candidate who you are, including your relation to the position. Be very clear about your role. Are you a hiring manager? A potential coworker in the same group, team, or department? Are you from another group, with whom the person would be cooperating closely? Are you a team member looking for team fit/ Or what? You know why the candidate is here – why are *you* here? He or she needs to know.

How long will we be at this? Where will we be? "This is conference room XYZ, folks will come to you (as opposed to my office), then I'll take you to ABC's office." Who will the candidate see next? And so on. Knowing what to expect goes a long way toward relaxing the candidate.

WHAT, WHEN

Resist the temptation to only tell him or her about the next hop in the chain. I've seen this done multiple times, and it's inexcusable. It generally happens in places that have had the experience of a candidate looking good on the resume, but not actually being as qualified as the resume implied. The firm cuts the candidate off after one or two people have done their interview sessions and there has been a decision that continuing would waste the company's time.

If you are doing first-round screenings, you shouldn't schedule more than one or two people in the first place. For that matter, if you don't have a phone-screen process that can keep this from happening, you are doing something wrong and need to fix your process. It's hard enough for the candidate to be out there looking. He or she does not need to feel that a door slammed in their face, instead of having one closed with some basic respect. Word gets around.

WHERE

Choose a space where you will not be interrupted. If your workplace does not usually schedule conference rooms, but instead does "first come, first served," you will probably want to post an abbreviated schedule on the door. This serves the dual purpose of reserving the room and letting people know how long you will be there. You don't need small meeting groups popping in every now and then to ask "will you be much longer?"

Try to get a space with a whiteboard or a blank flipchart pad (and markers!) for technical diagrams and off-the-cuff illustrated problems or war stories.

Consider environmental factors – if you are doing morning interviews in the south-facing conference room, the one with the minimal air conditioning, you will probably roast your poor candidate in his or her suit, not to mention yourself. Will one of you be squinting at the other through the bright sun, or does the room have blinds or shades? Just common sense, but keeping it in mind is a good thing.

How

Relax the candidate a bit. This is especially important for technical positions. Many folks who are very competent are better at typing than talking, or don't deal with humans as well as with solving problems.

If you are the first interviewer, or someone in the middle of a long string of interviewers, be sure to offer the candidate some basic hospitality. Ask if he or she needs to take a quick break to go "down the hall." See if a drink would be welcome. Many people are nervous about asking, or may refuse if you are not also having something. If the office break room is nearby, both of you could duck out to snag a soda or coffee to bring back to the conference room.

Warn him or her about any "tricks" you do in advance, which makes it more understandable and more of a game than a challenge.

Example: "I just want to let you know that some of the questions I'll be asking you have one right answer, some have multiple right answers, and some are open problems in the field that may not have an answer at all. I want to know how your mind works as much as what things you already know, since a lot of this position involves dealing with things that nobody really has working yet!"

Take notes during the interview. Mention to the candidate that you do this for all candidates, so that he or she won't think you are only taking unfavorable notes. Good topics for notes are things that impress you positively, things that you have questions about, things that strike you as strongly negative, and things that you want to keep in mind during the feedback process.

Hint: Take notes on the back of the cover sheet of the candidate's resume so that you won't have to keep flipping over your notes to ask about things on the resume. You did bring the resume, right? And you did read it before the interview?

Many folks who are very competent are better at typing than talking, or don't deal with humans as well as with solving problems.

“What’s the thing you’re proudest of that you’ve pulled off?”

Make sure there’s a “thank you.” Each interviewer should thank the candidate for his or her time, and find something on which to compliment them. It’s just basic business courtesy, and we sysadmins as a profession often lag behind on this one. “Well, thanks, uh, I guess somebody will call you when we know something” – c’mon, you can do a little better than that!

The last person to see the candidate on that day should hand them off to HR or the hiring manager for a goodbye, or should be someone polished enough to do a smooth goodbye. Let the candidate know that you appreciate their presence, that someone will be contacting them by such-and-such a date at the latest, and make sure that you have correct contact information.

AFTERWARD: THE FEEDBACK

Do an individual feedback session for each team member as well as a roundtable feedback session with everyone who is part of the interview process. Your team, like people in general, will often be more candid in private and may have opinions that they aren’t comfortable sharing with the group.

AND THE OTHER FEEDBACK

Be sure that the candidate hears from someone in HR or from the hiring manager that same day or early the next day. If you are very organized, you could skip this step because you’d have told the candidate at the “goodbye” part of the interview that he or she would be hearing from you by a particular day. After all, the feedback roundtable would already be on your calendars, and the interview slots for second round as well. Since most of us are not this organized, it is a good idea to call the candidate and reiterate your thanks for their time and establish a time when he or she will be contacted with definite news about next steps (or the lack of them).

A FEW FAVORITE QUESTIONS

When comparing notes on interview questions, a colleague shared this amusing tidbit: “At the end I usually like to ask if there is anything he or she would like to add which might influence my decision in any way. The most memorable response to date: ‘Well, I’m presently in the process of suing my last employer.’”

Yow! Well, favorite interview questions are a whole article in and of themselves, but here are some that I believe should be included in the secondary rounds of interviewing, maybe even in the first rounds for certain sorts of positions.

“What’s the thing you’re proudest of that you’ve pulled off?”

I like to phrase this one roughly that way, as it elicits a different and usually more interesting type of response than the dry “what do you feel has been your greatest accomplishment?” When did you beat the odds, climb the mountain, make the sun stop in the sky? Even if it wasn’t something that looked like a big deal from the outside – tell me about it! This is a really good way to find out more about the values and experience of the candidate in one big rambling package.

It’s also a good way for candidates to be able to talk about things they might have done outside the context of work – collaborated on an open-source project, set up email for their high school, pulled puppies out of a burning building. If someone talks about work-related stuff in response to this, and I feel that I have a good rapport with the candidate, I will usually ask if they’d share something outside the context of work in

this area. I always assure them that it is highly optional, but most people get too few opportunities to share their triumphs.

“Tell me about a situation that you wish you’d handled differently, and why – and with 20-20 hindsight, what you would have done instead.”

Everybody has these! Everybody! I can tell you about some doozies, believe me. For those of us who started on SunOS, we can all remember the first time we messed up the dynamic libraries, usually late at night, and how we found our way back to civilization some millennia later. I want to know that the candidate is comfortable enough to tell me about a learning experience, and smart enough to have learned from one. You’d be surprised at the number of folks who will describe a problem and say the way they wish they’d handled it was not to do it in the first place. For most problems, that’s not learning solutions, that’s learning avoidance behavior.

“What’s the worst technical situation you’ve ever seen someone else create, and how did it get that way?”

Rare indeed is the sysadmin or manager who has not seen something so hideous and crafty that it evoked a reaction of “must . . . fix . . . problem . . .” We’ve also stood by and watched helplessly as those outside our sphere of influence struggled with something we were not allowed to help fix, and felt that horrible empathy as defeat was once again snatched from the very jaws of victory. So, did we learn anything? War stories also build rapport – don’t be afraid to share one of yours, but perhaps it should be something from a previous job, and clearly labelled as such.

Last but not Least

Don’t give up too soon!

Another mistake often made is to narrow the search too quickly. Obviously you shouldn’t waste your people’s time, and good candidates usually go quickly. Still, the “bird in the hand” syndrome means that most inexperienced hiring managers don’t try to identify all qualifying candidates. Instead, we often see a rather futile cycle.

The first candidate who is fairly decent and has good team interaction is turned into the “best-fit candidate” by some sort of Procrustean revisionism on the job description or requirements page. A few months or weeks later, the disconnect between the original needs and the candidate becomes a problem for the team, the new employee, or both. In the worst-case scenario, the candidate search begins all over again. The easiest way to avoid this mistake is to keep doing first-round interviews even when you have promising candidates in the second-round stage.

Yes, I said “candidates,” since another frequent mistake is to limit the number of people you advance to the second round. With today’s abbreviated hiring cycles, often born of desperation, there can be an expectation that getting to a second round means the fix is in. Not by a long shot! Any truly qualified candidate who is a potential good match for the job should be advanced to the second-round stage.

Do not progress serially – seeing if one candidate “passes” to second round with the team and only contacting other candidates if he or she does not. Second round should progress in much the same fashion as the first round, with all of the candidates interviewing within the same two or three days. If there are two candidates who are both so sufficiently well regarded that it seems difficult to choose between them, you may want to bring those two in for a third round. Depending on your organization’s growth rate, you may also want to see if you can offer a position to both of them!

Don’t give up too soon!

the first miracle of management

by Steve Johnson

Steve Johnson has been a technical manager on and off for nearly two decades. At AT&T, he's best known for writing Yacc, Lint, and the Portable Compiler.

<scj@transmeta.com>



and Dusty White

Dusty White works as a management consultant in Silicon Valley, where she acts as a trainer, coach, and trouble shooter for technical companies.

<dustywhite@earthlink.net>



Those of you who have been asked to manage know that it's the kind of job that looks easy until you have to do it. Then you discover that all the "power" that seemed to be attached to the position has trickled through your fingers and disappeared; your group is persisting in doing what it wants to do rather than what you want it to do; and your boss is losing patience.

In the midst of all of this struggle, there are a few bright spots. One we have called the First Miracle of Management: There are people who love to do what you hate to do, and they are very, very good at it!

If you have trouble with corporate budgeting, there are people who are very good at this. If you hate making detailed plans, or preparing documentation, or organizing the department picnic, there are people who love to do this kind of thing.

Accepting the First Miracle just leaves you with just three things to do:

1. Make sure there are such people around.
2. Find them.
3. Get them to do the work.

The first task may be the hardest. Many organizations are kind of a monoculture – the hiring manager hires clones of himself or herself, so most people have exactly the same strengths and weaknesses of the manager. (After all, it's natural to hire someone like you because you tend to "like them.") So you may have a group with ten great coders and nobody who can document or plan. And it's usually quite clear in such a group that people were hired to code, period.

Luckily, people are pretty difficult to pigeonhole. Even in the most monocultural group, there is likely to be a spectrum of ability with respect to these "secondary" tasks, many of which are in fact crucial for the group's success. Of course, if the group was hired carefully with the idea of creating a collection of diverse, interlocking skills, then you may already have some excellent candidates.

So how do you find them? In a monoculture, these people may have gotten the message that they were hired to code, and only coding gets rewarded! In the worst case, they may get no credit for doing noncoding work, or even get criticized because their "productivity" is lower when they are doing noncoding tasks. But let's assume that the tight shoe is pinching so badly that the managers, and quite possibly the rest of the group, recognize that something must be done.

In many cases, you just need to let people know that the job needs doing, and some people will come to you (usually one-on-one) and volunteer. Many people who are good at coding nevertheless don't like to do it 60 hours a week, and they would welcome the chance to break up their other responsibilities with some planning or documentation or whatever. And remember the Miracle – they may even be really good at these jobs and like to do them.

So then the question is, how do you get them to do the work? In many cases, the answer is simple and elegant: offer to do something for them that you enjoy and are good at, and that they hate. Then they get to share in the Miracle as well. In other cases, even the most modest perks or recognition will work, because, remember, they basically like doing that job you hate. Sitting down and discussing what needs to be done can also do wonders – it may be that they like 75% of it, and the 25% that is left over doesn't bother you, so you can carve up the job differently and both be happy.

In the midst of all the “interesting” experiences a new manager faces, let's hope that she or he experiences the Miracle and learns how to use it.

the bookworm

BOOKS REVIEWED IN THIS COLUMN

UNIX SYSTEM ADMINISTRATION HANDBOOK, 3RD ED

EVI NEMETH ET AL.

Upper Saddle River, NJ: Prentice Hall PTR, 2001. Pp. 853. ISBN 0-13-020601-6.

THE FIRST COMPUTERS

RAUL ROJAS & ULF HASHAGEN, EDS.

Cambridge, MA: MIT Press, 2000. Pp. 457. ISBN 0-262-18197-5.

BEAUTIFUL CORPORATIONS

PAUL DICKINSON & NEIL SVENSEN

Upper Saddle River, NJ: Prentice Hall/Financial Times, 2000. Pp. 129. ISBN 0-273-64233-2.

ESSENTIAL XML

DON BOX ET AL.

Boston, MA: Addison-Wesley, 2000. Pp. 368. ISBN 0-201-70914-7.

XML POCKET REFERENCE

ROBERT ECKSTEIN

Sebastopol, CA: O'Reilly, 1999. Pp. 107. ISBN 1-56592-709-5.

THE XML COMPANION

NEIL BRADLEY

London: Addison-Wesley, 2000. Pp. 317. ISBN 0-201-67487-4.

by Peter H. Salus

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



<peter@matrix.net>

My "Top 10" list for 2000 is at the end of this column.

Third Time Around

As you faithful readers know, I generally avoid mentioning second (third, nth) editions. But last time there was the *Firewalls* book and now there's a third edition of the *UNIX System Administration Handbook*. Evi Nemeth and her collaborators have done the truly splendid job I've come to expect of them. In 1988 and in 1994 I saluted the "red book."

The third edition has turned purple. It has also added full coverage for Red Hat Linux. And covers IPv6. The first edition had a foreword by Dennis Ritchie. The second reprinted that, plus one by Kirk McKusick and Eric Allman. This edition reprints both of those *plus* one by Linus Torvalds. It's been the best book in the field for a dozen years.

Run out and buy your copy now!

History

In August 1998, the International Conference on the History of Computing was held at the Heinz Nixdorf Museums Forum in Paderborn, Germany. *The First Computers* contains two dozen papers, all but one presented in some form at the conference. They have all been extensively rewritten, and a superb introductory chapter by Michael R. Williams has been prefixed to them. The editors, Raul Rojas and Ulf

Hashagen, have added an informative preface.

The questions surrounding first computers as a concept are manifold. Reading this volume makes it clear that computer science arose simultaneously in several places. (Think of Priestley and Lavoisier in chemistry, or Darwin and Wallace in biology, or Newton and Leibniz in mathematics.)

This book outlines the trains of thought where computer architectures are concerned and where the various developments in the US, in Germany, and in Britain were concerned. The photos and diagrams are just wonderful. MIT Press has executed the production quite lovingly.

If there's any lack, it's still George Stibitz. Though his name appears in the table on p. 10, there's no essay on him nor his work at Bell Labs in the 1930s and 1940s.

Style

Another interesting and handsome volume is *Beautiful Corporations*. I hadn't been giving much thought to the manner in which companies construct a strong identity for themselves and their customers. Yet this is obviously the major function of marketing and PR firms. (I generally think of graphic design only in a Tufte sense.)

Dickinson and Svensen have really gotten me thinking about the driving force in our capitalist culture. Even if it isn't computing, it's important.

Marking It Up

SGML and HTML are giving way to XML. Box et al., which I reviewed last time, try to tell us that the Extensible Markup Language has turned into "universal duct tape" for software applications. Box, Skonnard, and Lam enabled me to acquire a genuine understanding of the inner workings of XML, as they present it. But they present it from a singularly slanted point of view. They see

XML as replacing Java and lots of other stuff. I'm not certain that I buy this Redmond point of view, yet. Box et al. reprint the latest W3C draft, which they state is "hopelessly out of date." Not to me.

A really splendid, but brief, version of XML's true abilities and purpose can be found in Eckstein's *XML Pocket Reference*. A part of O'Reilly's continuing pocket-sized series, the *XML Pocket Reference* is all that and much more. It introduces XML, gives a good intro to the definitions, and sits next to this workstation as I type. It's a gem!

In a different way, Bradley has given me a real insight into the way that XSL (Extensible Stylesheet Language) works. In fact, he also informed me about the limitations of the whole family of stylesheet standards.

I've recommended these two books to my Web crew.

The Year's Top Ten

I had a lot of trouble picking only ten books this year, and I ended up cheating, as #10 is a series. I guess I'll find coal in my shoe/stocking this year.

1. Zwicky, Chapman & Cooper, *Building Internet Firewalls* (O'Reilly)
2. Schneier, *Secrets and Lies* (Wiley)
3. Nemeth et al., *UNIX System Administration Handbook* (Prentice Hall PTR)
4. Callaghan, *NFS Illustrated* (Addison-Wesley)

5. Garfinkel, *Database Nation* (O'Reilly)
6. Smith, *[Incr Tcl] from the Ground Up* (Osborne/McGraw Hill)
7. Pawlan, *Essentials of the Java Programming Language* (Addison-Wesley)
8. Metcalfe, *Internet Collapses* (IDG Books)
9. Rojas & Hashagen, eds., *The First Computers* (MIT Press)
10. Loshin, ed., *Big Book of . . . RFCs* series (Morgan Kaufmann)

SAGE Elections

Elections for the SAGE Executive Committee for the 2001-2003 term are coming up soon. For the first time, voting will be conducted electronically. VoteHere.net has been selected to conduct the elections.

To be eligible to vote, you must be a SAGE member on December 1, 2000. Since voting will take place electronically, it is essential that your membership information is up to date, particularly your email address. To vote, you will need your membership number and password.

To verify and update your membership information, please go to:
<https://db.usenix.org/membership/update_member.html>.

Election notifications and candidates' statements will be available on the SAGE Web site (<<http://www.usenix.org/sage/election01/>>) by December 11, 2000.

Notifications and voting instructions will be sent via email to all current SAGE members. For those who choose to not submit their ballots electronically, a paper ballot option will be made available through the voting website.

To find out more about the candidates running for seats on the Executive Committee, please attend the Candidates' Forum being held on December 7, 2000 at LISA in New Orleans. Candidates' statements will also be available through the SAGE website.

For more information about SAGE governance, please see:
<<http://www.usenix.org/sage/official/>>.

SAGE 2000 Salary Survey

SAGE invites you to participate in the 2000 System Administrator Salary Survey at <http://www.usenix.org/sage/jobs/salary_survey/survey>.

This annual survey is part of SAGE's ongoing effort to advance the profession through information and advocacy. Results will be available in the coming months to SAGE members and to those who participated in the survey.

LISA Conference Attendees: Take the survey, and then stop by the Membership Booth for your thank you gift.

SAGE STG EXECUTIVE COMMITTEE

President:
Barb Dijker <barb@sage.org>
Vice-President:
Xev Gittler <xev@sage.org>
Secretary:
David Parter <parter@sage.org>
Treasurer:
Peg Schafer <peg@sage.org>
Members:
Geoff Halprin <geoff@sage.org>
Hal Miller <halm@sage.org>
Bruce Alan Wynn <wynn@sage.org>

SAGE SUPPORTING MEMBERS

Collective Technologies
Deer Run Associates
Electric Lightwave, Inc.
ESM Services, Inc.
GNAC, Inc.
Mentor Graphics Corp.
Microsoft Research
Motorola Australia Software Centre

New Riders Press
O'Reilly & Associates Inc.
Remedy Corporation
RIPE NCC
Sams Publishing
SysAdmin Magazine
Taos: The Sys Admin Company
Unix Guru Universe

USENIX MEMBER BENEFITS

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

FREE SUBSCRIPTION TO ;login:, the Association's magazine, published eight times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on security, Tcl, Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

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

ACCESS TO PAPERS from the USENIX Conferences online starting with 1993
<www.usenix.org/publications/library/index.html>.

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

OPTIONAL MEMBERSHIP in SAGE, the System Administrators Guild.

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/membership.html>>

OR CONTACT

<office@usenix.org>

Phone: 510 528 8649

USENIX news

Opportunities for Students

by **Bleu Castañeda**

Production Editor

<bleu@usenix.org>

USENIX and SAGE offer a variety of opportunities for students. Two of these programs are spotlighted here, the Student SysAdmin Internship program and the USENIX Student Stipend program. For more information about all of our student programs, please see our website at:

<<http://www.usenix.org/students/students.html>>.

Student SysAdmin Internship Program

Earlier this year, SAGE initiated the Student SysAdmin Internship Program. The purpose of the project is to encourage more students to acquire systems administration skills. University faculty or system administrators may be particularly interested in this program.

The program was born from the observation that many systems administrators have their first systems administration exposure while in college through on-the-job experience, not through coursework that may or may not be related to computer systems.

SAGE launched the program in June by funding four \$5,000 pilot projects enabling mentors to hire students to do systems administration tasks. For descriptions of the funded projects, please visit:

<<http://www.usenix.org/sage/projects/internship/index.html>>.

USENIX Student Stipend Program Report

USENIX offers many opportunities for students to get involved in Association activities. One way is the Student Stipend Program which makes it possible for students to attend USENIX conferences and workshops. This year, USENIX has given student stipend awards totaling \$94,830 to 123 students from 69 different institutions in the United States and abroad. Since the program's inception in 1990, USENIX has made 1,069 student stipend awards, totaling \$736,229 (an average of \$689 per student).

Student stipends cover travel, accommodations and registration fees for full-time students interested in attending USENIX conferences and workshops. The application process is short and easy-to-complete. Instructions, including the application and deadlines for specific events, can be found at:

<<http://www.usenix.org/students/stipend.html>>.

USENIX BOARD OF DIRECTORS

Communicate directly with the USENIX Board of Directors by writing to: <board@usenix.org>.

PRESIDENT:

Daniel Geer <geer@usenix.org>

VICE PRESIDENT

Andrew Hume <andrew@usenix.org>

SECRETARY:

Michael B. Jones <mike@usenix.org>

TREASURER:

Peter Honeyman <honey@usenix.org>

DIRECTORS:

John Gilmore <john@usenix.org>

Jon "maddog" Hall <maddog@usenix.org>

Marshall Kirk McKusick <kirk@usenix.org>

Avi Rubin <avi@usenix.org>

EXECUTIVE DIRECTOR:

Ellie Young <ellie@usenix.org>

CONFERENCES

Barbara Freel <barbara@usenix.org>

Registration/Logistics

email: <conference@usenix.org>

Telephone: 510 528 8649

FAX: 510 548 5738

Dana Geffner <display@usenix.org>

Exhibitions

Telephone: 831 457 8649

FAX: 831 457 8652

Daniel V. Klein <dvk@usenix.org>

Tutorials

Telephone: 412 422 0285

Bygone Times

by Peter H. Salus

USENIX Historian
<pete@Matrix.Net>

I didn't call this 20 Years Ago, nor 30 Years Ago, largely because coming at the close of 2000 (the end of a millennium in my numerical system), and having just returned from a conference, I am feeling nostalgic.

While I was unpacking my stuff in my new office, I came across some things that added to this feeling. One "document" was Mike Lesk's "Tbl - A Program to Format Tables" [Bell Laboratories Computing Science Technical Report #49, September 1976]. A second was "Software Tools Subsystem User's Guide" [Georgia Tech, GIT-ICS-79/07, September 1979]. The third was "The UNIX System," by Stephen R. Bourne [USENIX tutorial notes, "5/84" - from the Salt Lake City meeting, June 1984].

And then I went away.

I was at the 4th Atlanta Linux Showcase. While there I had several talks with Paul Manno. Paul was one of the Georgia Tech "kids" who were involved with the software tools. I recall meeting him for the first time in June 1986, in the Atlanta

Hilton, as Paul was one of the hosts of the Summer 1986 USENIX Conference.

It was a really interesting conference: Mike O'Dell was the program chair; Jon Bentley was the keynote speaker; Peter Langston and Mike Hawley gave music papers; Rick Rashid and the CMU crowd gave the first Mach paper; Amnon Barak gave the first MOSIX paper; etc. And, of course, there was the exhibition of the USENIX synchronized swim team.

Sigh.

On a totally different topic, some of you may recall my discussion of anniversaries. I got a note from Tom Limoncelli at Bell Labs which actually resolves the entire question. He wrote:

I'm catching up on my *login*s and read the June issue where you wondered about the logic behind having the 10th anniversary of USENIX celebrated in 1985 and I didn't see any follow-up in the issues that followed. My theory is that they were using octal and celebrating the name-change to USENIX that was in July 1977 which was, in octal, 10 years prior.

As one of those young whippersnappers who started using computers when hexadecimal had become popular, I'm surprised that I noticed this at

all. Maybe it's because I was just dealing with a user who thought it was a bug that "telnet 137.100.050.099" didn't get to the right machine.

I wonder what computers will be like when we celebrate our base64 10-year anniversary in 2041.

You're right, Tom. I'm horrified that I didn't realize it.

Season's greetings and a good new millennium to those of you who can count in decimal.

[We don't often reproduce news releases, but this one seemed too good to pass by. ed.]

BIND 9 Authored by Nominum Development Team Now Available on Internet Software Consortium Site

The Internet Software Consortium (ISC) has announced the release of BIND 9, written by Nominum, Inc. under an ISC outsourcing contract. BIND, an acronym for Berkeley Internet Name Domain, is the most commonly used domain name server on the Internet and implements the Domain Name System (DNS) suite of protocols. DNS enables virtually all internetworking applications such as e-mail, Web browsers and file transfers. Available as Open Source from the

MEMBERSHIP

Telephone: 510 528 8649
Email: <office@usenix.org>

PUBLICATIONS/WEB SITE

<<http://www.usenix.org>>
Jane-Ellen Long <jel@usenix.org>
Telephone: 510 528 8649

USENIX SUPPORTING MEMBERS

Addison-Wesley
Earthlink Network
Edgix
Interhack Corporation
Interliant
Lucent Technologies
Microsoft Research
Motorola Australia Software Centre
Nimrod AS

O'Reilly & Associates Inc.
Sams Publishing
Sendmail, Inc.
Smart Storage, Inc.
Sun Microsystems, Inc.
Sybase, Inc.
Syntax, Inc.
Taos: The Sys Admin Company
UUNET Technologies, Inc.

Internet Software Consortium, BIND 9 is the world's first DNS implementation to fully support IPv6 and the DNS security enhancements specified by the Internet Engineering Task Force (IETF) standards body in RFC 2535. Russ Mundy, Manager of Network Security Research for NAI Labs, affirms the importance of BIND 9's security features, "As a leading provider of security solutions, Network Associates is pleased to have contributed to the security-related design and development of BIND 9. The improved security and performance capabilities of BIND 9 help to secure the name system for the Internet. Our researchers at NAI Labs continue to work with the ISC, the Internet Engineering Task Force and key network operators to help bring long-needed security to critical elements of the DNS." Based on a multi-processor scalable, multi-threaded architecture, BIND 9 is a complete rewrite of BIND, which was originally written in 1983 at the University of California at Berkeley. This latest release features modularized code for security auditability, use of programming by contract development paradigm for internal consistency checking and much greater RFC conformance while maintaining a large degree of backwards compatibility with earlier version of BIND.

David R. Conrad, Executive Director of the Internet Software Consortium, commends the enhancements of the much-anticipated BIND 9 release: "I want to thank all the members of the Open Source community who contributed to this important effort. The newest implementation of BIND has been completely re-architected, with clean interfaces between internal modules and a streamlined architecture that is light-years ahead of previous versions. Nominum's development team took the time to build BIND 9 from the bottom up, and their hard work has really paid off, resulting in extraordinary speed, scalability and security." The development team at

Nominum has been developing BIND and DHCP (Dynamic Host Configuration Protocol) software for the Internet Software Consortium as well as providing support, consulting, training and custom development to a global user base. The ISC is a non-profit organization dedicated to developing and maintaining quality Open Source reference implementations of core Internet protocols. BIND version 9 development was underwritten, in part, by the following organizations:

- Compaq Computer Corporation
- Hewlett Packard
- IBM
- IPWorks, Inc.
- Network Associates, Inc.
- Silicon Graphics, Inc.
- Stichting NLNet
- Sun Microsystems, Inc.
- U.S. Defense Information Systems Agency (DISA)
- USENIX Association
- Verisign, Inc.

"USENIX is always supportive of projects like Bind 9," says Andrew Hume, Vice President of the USENIX Association. "It directly helps our members who are system and network administrators, and facilitates our other

members who use the Internet to do their research or to build their products. We also appreciate that Bind 9, like nearly all the software that the USENIX community has historically used, is Open Software."

To download a copy of BIND 9, users should go to

<http://www.isc.org/products/BIND/bind9.html> and follow the instructions outlined therein. For BIND 9 documentation, including system requirements, configuration references, troubleshooting and security considerations, please visit

<http://www.nominum.com/resources/Bv9ARM-091200.pdf>.

Statement of Ownership, Management, and Circulation, 10/23/00

Title: ;login: Pub. No. 0008-334. Frequency: Bimonthly plus July and November. Subscription price \$60 individuals and institutions. Office of publication: USENIX Association, 2560 9th Street, Suite 215, Berkeley, Alameda County, CA 94710. Headquarters of Publication: Same. Publisher: USENIX Association, 2560 9th Street, Suite 215, Berkeley, CA 94710. Editors: Rob Kolstad & Tina Darmohray. Managing Editor: Jane-Ellen Long, all located at office of publication. Owner: USENIX Association. The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes has not changed during the preceding 12 months.

Extent and nature of circulation	Average no. of copies of each issue in preceding 12 months	Actual no. of copies of single issue published nearest to filing
A. Total no. of copies	12,379	12,450
B. Paid and/or Requested Circulation		
Sales through Dealers	0	0
Mail Subscribers	10,942	10,703
C. Total Paid and/or Requested Circulation	10,942	10,703
D. Free Distribution by Mail	72	86
E. Free Distribution Outside the Mail	797	1,220
F. Total Free Distribution	869	1,306
G. Total Distribution	11,811	12,009
H. Copies Not Distributed	568	441
I. Total	12,379	12,450
Percent Paid and/or Requested Circulation	88%	86%

I certify that the statements made by me above are correct and complete.
Ellie Young, Executive Director