NOVEMBER 1999

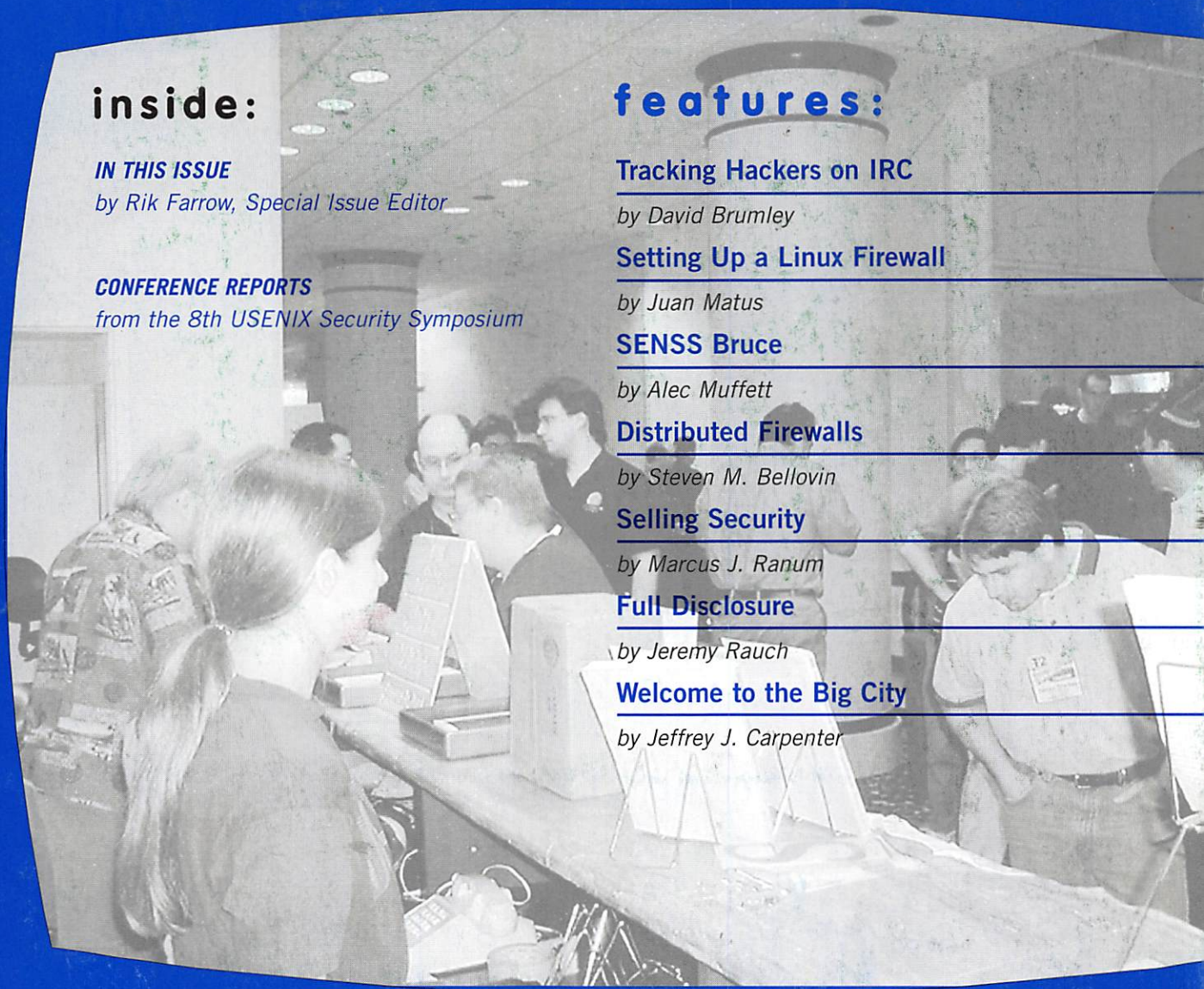
;login:

The USENIX Association Magazine

Special Issue on Security

SPECIAL ISSUE

Security



inside:

features:

USENIX The Advanced Computing Systems Association

Events

USENIX *Upcoming* Events

NordU 2000—2nd Nordic EurOpen/USENIX Conference

February 8-11, 2000
Malmö, Sweden
Web site: http://www.nordu.org/NordU2000/

7th USENIX Tcl/Tk Conference (Tcl/2k)

February 14-18, 2000
Marriott Hotel, Austin, Texas, USA
Web site: http://www.usenix.org/events/tcl2k

Workshop on Applications of Embedded Systems

March 20-22, 2000
San Francisco, California, USA
Web site: http://www.usenix.org/events/es2000
Submissions due: November 15, 1999

SANS 2000—9th International Conference on System Administration, Networking, and Security

Co-sponsored by the SANS Institute and SAGE
March 21-28, 2000
Orlando, Florida, USA
Web site: http://www.sans.org

SANE 2000—2nd International System Administration and Networking Conference

Organized by NLUUG, co-sponsored by USENIX and Stichting NLnet
May 22-25, 2000
Maastricht, The Netherlands
Web site: http://www.nluug.nl/events/sane2000/
Submissions due: November 1, 1999

2000 USENIX Annual Technical Conference

June 18-23, 2000
San Diego Marriott Hotel & Marina, San Diego, California, USA
Web site: http://www.usenix.org/events/usenix2000
Submissions due: November 29, 1999

3rd Large Installation System Administration of Windows NT/2000 Conference (LISA-NT 2000)

July 30-August 2, 2000
Madison Renaissance Hotel, Seattle, Washington, USA
Web site: http://www.usenix.org/events/lisa-nt2000
Submissions due: February 16, 2000

4th USENIX Windows Systems Symposium

August 3-4, 2000
Madison Renaissance Hotel, Seattle, Washington, USA
Web site: http://www.usenix.org/events/usenix-win2000
Submissions due: February 11, 2000

9th USENIX Security Symposium

August 14-17, 2000
Denver Marriott City Center, Denver, Colorado, USA
Web site: http://www.usenix.org/events/sec2000
Submissions due: February 10, 2000

4th Symposium on Operating System Design & Implementation (OSDI 2000)

Co-sponsored by IEEE TCOS and ACM SIGOPS
October 23-25, 2000
Paradise Point Resort, San Diego, California, USA
Web site: http://www.usenix.org/events/osdi2000
Submissions due: April 25, 2000

6th USENIX Conference on Object-Oriented Technologies and Systems

January 29-February 2, 2001
San Antonio, Texas, USA
Web site: http://www.usenix.org/events/coots01
Submissions due: July 27, 2000

For a complete list of future USENIX events, access *http://www.usenix.org/events*

contents

Cover: The USENIX Booth at the Security Symposium

;login:

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

;login: (ISSN 1044-6397; USPS 0008-334) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $80 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

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

Editorial Staff

Editor:
Rob Kolstad <kolstad@usenix.org>

SAGE News and Features Editor:
Tina Darmohray <tmd@usenix.org>

Standards Report Editor:
Nick Stoughton <nick@usenix.org>

Managing Editor:
Jane-Ellen Long <jel@usenix.org>

Copy Editor:
Eileen Cohen

Proofreader:
Kay Keppler

Designer:
Vinje Design

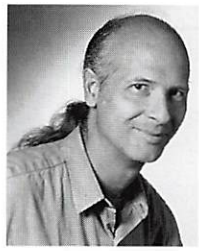
Typesetter:
Festina Lente

Membership and Publications

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

The closing dates for submission to the next two issues of *;login:* are December 1, 1999, and February 2, 2000.

in this issue . . .



by Rik Farrow
Special Issue Editor
<rik@spirit.com>

Welcome to the second Security Special Edition of *;login:*. Ellie Young initiated these special editions, and this is the fourth that I have had the honor of editing.

Being editor means, among other things. that you get to read through everything in the magazine several times. Once or more when you first get a submission (depending on whether the article needs revisions), and then again for the page proofs (to be certain that nothing strange has crept in during the editing/typesetting process). I am happy to say that I enjoyed reading through this issue again.

The summaries were the hardest part for me to reread, because I had already read them several times before they were turned in. In this case, I encouraged the summarizers to feel free to write longer summaries than usual as we had enough space for them. What you will find in the summaries are descriptions of the paper presentations of the Eighth USENIX Security Symposium, which took place in Washington, D.C., this past August 23-26, during a spell of surprisingly mild Washington weather.

USENIX conferences are places for learning. The summaries not only outline the contents of each paper presented but also attempt to depict the atmosphere during the presentation, as well as information that is not contained in the papers themselves – for example, questions from the audience, comments that induced laughter (or snickering), as well as the impressions of the summarizers themselves. If you are really interested in absorbing the contents of a USENIX session, writing the summaries is a good opportunity to do so.

The Invited Talks track was both educational and interesting. Ed Felton was not able to speak because of a sudden flu, but I expect that he would have shared insights that were released in a press release the following week. New tricks for exploiting ActiveX were the topic of Richard Smith's talk, and those very exploits were released the following week by his friend Georgi Guniski (see the acknowledgment to him in Microsoft Security Advisory MS99-040).

Steve Bellovin's talk about why cryptography has become important in the Internet covers both the strengths and the failures of the uses of cryptography. In the features section, Steve's paper about distributed firewalls builds on features of IPSec to create a new generation of firewall.

Susan Landau's talk about US crypto policy turned out to be prophetic indeed: a few weeks later, a presidential edict loosened the restrictions on export policy. This change may affect SENSS Bruce, a freely distributable Sun tool for distributing patches and security scripts as well as collecting output from these scripts in a hierarchical manner (see Alec Muffett's article). Both Steve Bellovin and Alec Muffett presented WIPs that turned into articles in this issue.

On the practical side, David Brumley, a member of the Stanford University Network Security Team, explains the importance of IRC in tracking down hackers. Setting up bots and relays is an important goal for hackers, and David clearly explains what you can look for in eggdrop (a popular UNIX bot) configuration files, as well as how IRC can be used to collect information about hackers.

No, the article about configuring a Linux firewall was not written by *the* Juan Matus; it's just that its author prefers to remain anonymous. There are, of course, other ways of setting up UNIX systems as firewalls. I am interested in receiving proposals for articles about setting up firewalls for laptops and on other Linux tricks, as well as for doing this with BSD.

Opinion

Then there is opinion. In this arena, we have Marcus Ranum, Jeremy Rauch, and Jeffrey Carpenter of CERT. Marcus writes about a different sort of intrusion detection (Marcus presented an IT about burglar alarms that you will find in the summaries and on his Web site). When someone sends out a press release about a new security problem, is that a service, or is it marketing hype? Jeremy Rauch, of SecurityFocus, takes an orthogonal tack on a similar subject by defending full disclosure. SecurityFocus has taken over archiving of Bugtraq from geek-girl and plans to become an important security information resource. From what I have seen so far, this site will be one to bookmark whether you are trying to configure IIS or a Cisco router.

Jeff Carpenter has written a position piece, describing a bit of the history of CERT as well as its current mission. CERT/CC's job is not to help handle incidents, but, rather, to be a single point for collecting trends about current incidents, as well as a reporting point for advisories about attacks that have become common enough to warrant such attention. I know that CERT's mission has often been misunderstood, which is why I offered this chance to clear up CERT's role in the world of Internet security.

Being There

My intent is that this edition is the next best thing to having been at the Symposium. Of course, you can catch only glimpses of what actually went on there, and perforce you miss the hall action.

Wiestse Venema was there. Wietse and Dan Farmer had just given another of their free seminars on UNIX security, this time at IBM's Watson Research Labs in New Jersey. All you had to do was send him email, get permission, then try to find a nearby hotel in a part of New Jersey where there are almost no hotels. I was unable to convince Wietse to write for this edition, but he did mention that he would put the slides of the class (about UNIX forensics) up on his Web site (<http://www.porcupine.org>).

Peter Neumann, who replaced the keynote speaker at the last minute, gave a wonderful talk about the base problem with computer and network security. But he also revealed himself as an expert on Tom Lehrer songs, playing on a grand piano for members of the Program Committee and getting quite a number of people to sing along.

And Peter Honeyman again revealed his Socratic soul. As anyone who has attended a number of USENIX conferences can tell you, Peter often demonstrates his ability to ask truly penetrating questions about a paper. Seeing Peter standing at the microphone can send premonitory chills down a speaker's back.

At the Symposium, Peter's most astonishing feat was to leap up to the microphone at the end of a paper that he himself had co-written and begin grilling his own co-author. He later explained that these questions had just occurred to him as he heard the paper presented.

Of course, the best thing would have been to be there. I hope that you will find this edition full of useful information, and I welcome your comments and suggestions for future articles about security that can appear in *;login:*.

conference reports

This issue's reports focus on on the 8th USENIX Security Symposium held in Washington, D.C., August 23-26, 1999.

Our thanks to the summarizers:

Michael J. Covington
<Michael.Covington@cc.gatech.edu>

Rik Farrow
<rik@spirit.com>

Kevin Fu
<fubob@mit.edu>

Matt Heavner
<heavner@gi.alaska.edu>

Ping Liu
<pingliu@cs.uiuc.edu>

Patrick McDaniel
<pdmcdan@eecs.umich.edu>

Jim Simpson
<freyjs@monkey.org>

8th USENIX Security Symposium

WASHINGTON, D.C.
August 23-26, 1999

KEYNOTE SPEECH: EXPERIENCE IS THE BEST TEACHER

Peter G. Neumann, SRI International

Summary by Kevin Fu

Peter Neumann of SRI International substituted for Taher Elgamal as the keynote speaker. Most of the talk dealt with issues such as security, survivability, reliability, and predictable behavior. Neumann has many claims to fame, including the moderation of <comp.risks> and a 1965 publication on filesystem access control in Multics. Neumann used stories, quotations, and clever puns to discuss principles of good software engineering.

Past efforts fundamental to software engineering include Multics, T.H.E. system, domains-of-protection principles, confined execution, PSOS (a provably secure operating system by PGN), and isolated kernels. But most existing commercial systems are fundamentally inadequate with respect to reliability and security. Survivability is also not well understood. Unfortunately, not much good research is finding its way into products. Developers ignore the risks, and the same old flaws appear over and over. For instance, eight of thirteen CERT reports this past year resulted from buffer overflows. Programming languages do little to prevent security problems. Few systems are easy to evolve.

Neumann challenged the old adage of KISS (Keep It Simple, Stupid). He argued that such advice does not work for extremely complex systems. He also disagreed with the "build one to throw away" principle of Brooks, because this encourages companies to perform beta testing with normal users.

There remains much to be learned from past mistakes; these problems have multiple dimensions. However, Neumann reasoned that we do not learn much from failures because the builders tend to move on to something else or build another system with the same problems.

The insider problem is largely ignored. This is a difficult problem to solve because the insider is already authorized. In defense of Wen Ho Lee, Neumann referenced the Los Alamos National Laboratory incident as an example.

Neumann recommended that developers specify general requirements, create stronger protocols, use good cryptographic infrastructures, design for interoperability, force commercial developers to do the right thing (maybe with open source), perform network monitoring and analysis, and find a good way to integrate systems.

The Q/A session consisted mainly of additional stories from the audience. Matt Blaze asked the rhetorical question, "Why in poorly designed systems does everything become part of the trusted computing base? Why limit [this characterization] to only poorly designed systems?"

John Ioannidis explained that Neumann is not "preaching to the choir" but to "lots of soloists." People do not listen to the soloists. Ioannidis argued that education of users is just as important as proper engineering practice. Otherwise, uneducated users will simply bring in the Trojan horse. The discussion led to an

insane number of analogies about singing and security.

Causing the crowd to erupt with laughter, Greg Rose said it's amazing that we can build great flight simulators, but we can't build good air traffic control systems.

Dan Geer quizzed Neumann on the difference between brittle and fragile. Neumann responded that brittle means an object breaks when tapped a bit; fragile means the object will probably fall off the shelf. In Multics, a failure in ring 0 would cause the whole system to collapse. A failure in ring 1 would only cause the process to crash. In ring 2, a failure might only generate an error message. The rings contained the error.

What is Neumann's final prognosis? All reports conclude that the state of practice is not good. Risks abound. For related information, including a pointer to the Risks archive, see <http://www.csl.sri.com/neumann/>.

Good quotation from the talk: "Too many people think security is a cookbook thing. Especially the government."

REFEREED PAPERS

Session: PDAs

Summaries by Kevin Fu

The Design and Analysis of Graphical Passwords

BEST STUDENT PAPER & BEST PAPER

Ian Jermyn, New York University; Alain Mayer, Fabian Monrose, Michael K. Reiter, Bell Labs, Lucent Technologies; Aviel Rubin, AT&T Labs – Research

Fabian Monrose analyzed the security of graphical passwords and proposed two graphical-password schemes that could achieve better security than textual passwords. This paper won both the best student paper award and the best overall paper award at the symposium.

Even though textual passwords can fall vulnerable to dictionary attacks, passwords still serve as the dominant authentication mechanism today. There's convincing evidence that graphical passwords, on the other hand, have better memorability and reveal less information about their distribution.

Results from cognitive science show that people can remember pictures much better than words. Combining this with the commonly found Personal Digital Assistant (PDA) allows new graphical input capabilities. Graphical schemes can decouple position from the temporal order in which a password is entered.

Monrose listed three desired properties of graphical passwords. First, they must be at least as difficult to search exhaustively as traditional passwords. Second, keys must not be stored as cleartext. Third, graphical passwords need to be repeatable and memorable.

After building a strawman scheme by which graphical passwords can emulate textual passwords, Monrose described a second scheme, dubbed Draw-a-secret. It takes as input a picture and the order of its drawing. This scheme keeps track of boundary crossings and pen lifts from the screen. This information then passes though a hash function to produce a raw bit string as a password.

Monrose pointed out that there is as yet no characterization of the distribution of graphical passwords. This means one picture is not known to be more likely than another as a password. On the other hand, textual passwords have well-known distributions related to dictionaries.

Rather than focus on the unrealistic upper bound of graphical-password choices, Monrose analyzed a minimum bound by creating a grammar (similar to LOGO) that describes common ways to enter a graphical password. Evidence indicates that by further assigning complexities to the terminals, graphical passwords with short complexities (pen strokes) can surpass the power of textual passwords.

Peter Honeyman went into an evil thesis-committee mode, shaking the earth with many difficult questions. Assuming at least 60 bits of entropy are necessary to protect confidential information, Honeyman questioned whether five pen strokes in a 5x5 grid is enough. Monrose did not answer the question directly.

Another audience member asked what would be the typical graphical password (a smiley face, picture of a dog, etc.). Suggesting that attacks similar to dictionary attacks may exist, the audience member asked how to classify what is memorable. Monrose explained that his team did not have enough experience to characterize distributions.

Another audience member asked for anecdotal evidence about memorability. Monrose explained that in practice, a 6x6 grid results in poor memorability. It's simply too fine-grained for the average user to repeatedly enter a graphical password correctly. The 5x5 grid creates a good balance between security and memorability.

Peter Neumann pointed out that written Chinese has a known keystroke order that may impose a distribution. Graphical passwords may pose a challenge for writers of Chinese. While a native writer of Chinese confirmed Neumann's point, she believes graphical passwords have merit.

Asked about incorporating timing and pressure for entropy, Monrose replied that it has not been considered. Neumann added that his group found pen pressure useless as a source of entropy.

While there is no concrete proof that graphical passwords are stronger than textual passwords, Monrose gave convincing evidence that graphical passwords stand a good chance and at least deserve further analysis. In the future, Monrose hopes to better classify memorability of graphical passwords. For source code and further information, see <http://cs.nyu.edu/fabian/pilot/>.

Hand-Held Computers Can Be Better Smart Cards

Dirk Balfanz and Edward W. Felten, Princeton University

Dirk Balfanz proposed that a PDA could serve as a better smartcard. By implementing a PKCS#11 plug-in for Netscape Communicator and the 3Com PalmPilot, Balfanz reduced the number of components in the Trusted Computing Base (TCB). The trusted part of the system remains only on the Pilot. Smartcards usually do not have a trusted authentication path to the user. The user often communicates a secret PIN to the smartcard via a PC and smartcard reader. This places the PC and smartcard reader in the TCB! Had any malicious software existed on the PC, it could have easily obtained digital signatures of bogus data. Of course, a user interface or warning light on the smartcard could prevent such malicious use. Unfortunately, most smartcards do not have a user interface.

The implementation of PilotKey currently works for Linux, Win9X, Solaris, and Windows NT in concert with Netscape Communicator and the Pilot. Preliminary results show that a 512-bit RSA signature takes about five seconds, and key generation about two to three minutes. However, a 1024-bit RSA signature takes 25 seconds and a whopping 30 minutes for key generation. Balfanz pointed out that the Pilot uses something like a 16MHz processor. He expects the processor to speed up in the future.

The PilotKey program implements the PKCS#11 functions for the Pilot. The program enables a user to provide randomness via the user interface. Because PilotKey works directly with the user, the PC does not see the PIN. Finally, the program lets the user choose whether or not to send decrypted messages back to the PC. Incidentally, PilotKey cannot securely handle email attachments (unless you can perform base64 decoding in your head).

Alas, the Pilot does exhibit some problems acting as a smartcard. First, it is not tamper-resistant, and the operating system does not provide memory isolation among programs. Hence, one should not use untrusted software on the Pilot in conjunction with PilotKey. Second, it is important not to lose the Pilot. While the secret information is password-protected, this offers only limited protection. Third, the PilotKey is not appropriate for "owner-is-enemy" applications. For instance, a program keeping track of a cash balance is inappropriate. The Pilot owner could easily change the balance. Finally, HotSyncing could reveal secrets to the host computer. Balfanz claimed these problems are not show stoppers. Inciting a few chuckles, he explained that you "just need a secure OS on the Pilot." In the paper, Balfanz makes some good suggestions on how to secure the Pilot OS.

Peter Honeyman began the inquisition with a hail of questions. Another audience participant reminded the rest of the crowd to turn off the Pilot's IR when not in use. One questioner asked why not just fix the OS on the PC if we can fix the OS on Pilot as suggested. Balfanz admitted this is the same problem, but fixing the OS on the PC is not any easier.

A participant suggested that splitting trust seems simply to push the problem down the line. People will want more features (e.g., signed Excel spreadsheets). Balfanz explained that shifting trust to the PDA comes at the expense of usability. Another participant argued that removing all the software from a Pilot results in a relatively expensive smartcard.

Questioned about Europe's desire for autonomous smartcard readers and smartcards with displays, Balfanz responded that he would not trust the ATM or the reader. Bruce Schneier wrote a similar paper on splitting trust (<http://www.usenix.org/publications/library/proceedings/smartcard99/schneier.html>) for the first USENIX Workshop on Smartcard Technology, held last May.

For more information on PilotKey, follow up with <balfanz@cs.princeton.edu>.

Offline Delegation

Arne Helme and Tage Stabell-Kulø, University of Tromsø, Norway

Arne Helme explained mechanisms for offline delegation of access rights to files in the context of a distributed File Repository (FR). In this model, each user has her own file repository but would like to share files. Offline delegation refers to delegating authority without network connectivity. For instance, one could use verbal delegation.

PDAs challenge centralized security models, provide users with personal TCBs, and support the construction of "delegation certificates." This meshes well with the design criteria for offline delegation: delegation should not allow impersonation; credentials must form valid and meaningful access rights; and the authority granted in a certificate should not be transferable or valid for multiple use.

The implementation consists of an access-request protocol and a prototype for the 3Com PalmPilot. The software contains a parser/generator for SDSI-like certificates and a short digital-signature scheme using elliptic-curve cryptography.

The access-request protocol (analyzed with BAN logic in the paper) describes the process of creating, delegating, and using offline certificates. The delegator must read 16 four-digit hexadecimal numbers to convey an offline certificate (e.g., by phone). Helme's Pilot software allows the delegatee to quickly receive these numbers via a convenient user interface.

Helme clarified that certificates in this scheme are valid for only one use. Servers keep a replay cache. One problem with the current signature scheme is that the FR cannot distinguish between two dif-

ferent files that had the same filename at different times.

Helme concluded that offline delegation is a natural extension of services to the File Repository, that PDAs support verbal delegation, and that performance is satisfactory. For more information, contact <arne@acm.org>.

Session: Cages

Summaries by Michael J. Covington

As was mentioned numerous times throughout the symposium, a critical component to building survivable systems is the ability to integrate untrusted information "pieces" – from hardware to mobile code and downloaded programs – into a trusted computing base. This session focused on building protection mechanisms for systems that would enable users to maintain security in their environment, while also being able to benefit from the use of an untrusted component. The three papers presented in this session discussed the design and implementation of prototype systems that provide a "caged" environment in which users are protected and enabled to proceed with their work in a safe manner.

Vaulted VPN: Compartmented Virtual Private Networks on Trusted Operating Systems

Tse-Huong Choo, Hewlett-Packard Laboratories

Tse-Huong Choo described the design and architecture of a software-based IPSec product named Vaulted VPN. The motivation behind the development of Vaulted VPN was to incorporate IPSec support into a trusted operating system. The move to a trusted OS was based on experience with conventional VPNs that have repeatedly failed because of security "hotspots" inherent in the operating system itself. By starting with a trusted foundation, Choo claims that security, performance, and overall robustness in the system are improved. More specifically, the trusted OS offers features such as sensitivity labeling and mandatory access control, while adhering to the principles of least privilege – all of which support the building of a compartmentalized IPSec implementation.

The system discussed in this talk is actually implemented as an IPSec VPN that consists of a series of compartmented, concurrently executing IPSec stacks. After briefly discussing some design alternatives, Choo presented the Vaulted VPN architecture, providing detailed descriptions of how packets move through the redesigned stack. In addition, he touched briefly on the topic of key management in the Vaulted VPN system.

One critical design feature, according to Choo, is the ability to have various components in the system run as a non-root user. By keeping the IPSec stack(s) in a separate compartment and running them in an environment that is stripped of most privileges, a single security failure will most often not lead to another. In addition, message channels are protected by a combination of both MAC (Mandatory Access Control) and DAC (Discretionary Access Control) – a feature that is not present on many standard operating systems today. Choo concluded by revisiting the benefits of a trusted OS and by commenting on the security gains achieved through compartmentalized designs.

Enforcing Well-Formed and Partially Formed Transactions for UNIX

Dean Povey, Queensland University of Technology

Dean Povey opened his presentation with a simple quote: "Sometimes it sucks to be a 'user.'" He proceeded to explain that although security is a critical component of information systems, users are often frustrated because they are not given sufficient rights to accomplish tasks assigned to them. With this problem as his motivation, Dean described an optimistic access-control system. By operating in a relatively trusted environment where the user population consisted of researchers and system administrators, Dean's system makes a base assumption that accesses are legitimate, but allows audit and recovery of the system when they are not.

There are three critical components to maintaining the security and stability of the environment in an optimistic system. First, there must be audit mechanisms in place to track who performs what functions on the system. There also must be accountability built into the system so that users who abuse their "extended rights" are held accountable. Finally, there must be a recovery mechanism that is capable of returning the system to a valid state should something go wrong.

Dean detailed `tudo` (Trusted User Do), an application designed to enforce both well-formed and partially formed transactions in a UNIX operating system. Based on `sudo`, `tudo` supports fine-grained access control of files and directories and also provides the logging and recovery features necessary to support well-formed and partially formed transactions. `tudo` incorporates its own access-control mechanisms, as well as recovery features that are being enhanced to provide greater stability and increased functionality.

The `tudo` prototype is available from <http://security.dstc.edu.au/projects/tudo>. This proof-of-concept implementation demonstrates how a reference monitor can be constructed using system call tracing facilities that enforce both well-formed and partially formed transactions.

Synthesizing Fast Intrusion Prevention/Detection Systems from High-Level Specifications

R. Sekar and U. Uppuluri, State University of New York at Stony Brook

The work described in this presentation was motivated by the building of survivable information systems and the construction of intrusion-detection systems that are able to isolate intrusions before they impact system performance or functionality. Sekar discussed the design and implementation of a system that allows users to specify acceptable patterns of system calls. Essentially, the runtime environment intercepts system calls and checks them against a set of specifications, disallowing or otherwise modifying those calls that deviate from the specifications. By intercepting calls before they reach the kernel, the system is capable of reacting before any damage-causing system call is executed.

Sekar presented samples of the specification language and provided an illustrated example in which he prepared some sample specifications for an FTP server. A critical component to his implementation was the development of a new, low-overhead algorithm for matching runtime behaviors against specifications. Surprisingly, the algorithm uses, in most cases, a constant amount of time per system call intercepted and it uses a constant amount of storage. Overall, experiments have demonstrated that the analysis of each system call contributes less than 5% overhead to the system.

Session: Keys

Summaries by Kevin Fu

Building Intrusion-Tolerant Applications

Thomas Wu, Michael Malkin, and Dan Boneh, Stanford University

Tom Wu discussed how applications can store and use private keys in an intrusion-tolerant manner. As a side benefit, Wu's methods also provide high availability of private keys. Existing public-key architectures rely on nonshared keys or split-key storage. The private key is reconstructed, creating a single point of failure. Methods in Intrusion Tolerance via Threshold Cryptography (ITTC) create no single point of attack because private keys are never reconstructed.

In Shamir's secret sharing scheme, a dealer generates a key and splits it into several shares for trustees. The trustees can later reconstruct the original key at a central location. In this scheme a dealer sees the original key and all the shares, creating a single point of failure. Compromise to the dealer's memory results in total disclosure of the key. Wu uses an intrusion-tolerant scheme that is not vulnerable to such a single point of failure. He employs methods, based on Boneh and Franklin, to generate a private key already in a shared form. To create shares, Wu uses an idea of Frankel's. Share servers apply the share to an operation (e.g., sign, decrypt, encrypt) rather than give the share to the application as if it were a dealer. Without a threshold number of results from the share servers, an adversary cannot reconstruct the results of a private-key operation. SSL protects the communication between the application and share servers. In order to break the ITTC scheme, an attacker must compromise multiple systems in a short period of time. One can also identify malicious share servers.

Wu's implementation adds intrusion tolerance to a Certificate Authority and Apache Web server. Integration of ITTC was trivial with the OpenSSL code. By relying on load balancing and multiple threads for parallelism, the ITTC adds only a 17% drop in throughput and 24% drop in latency when compared to non-ITTC RSA. Wu pointed out that this tolerable slowdown is insignificant considering the overhead of SSL session establishment.

Wu concluded that ITTC improves security and reliability for servers and CAs. Meanwhile, the performance impact is minimal. An audience member asked about the security consequences of an intruder obtaining a core file from such an intrusion-tolerant Web server. Wu happily replied that the intruder could retrieve only the result of a single decryption, not the private key of the Web server. At no single point is the key entirely reconstructed.

For more information, see <http://www.stanford.edu/~dabo/ITTC/>.

Brute Force Attack on UNIX Passwords with SIMD Computer

Gershon Kedem and Yuriko Ishihara, Duke University

Gershon Kedem gave an overview of brute-force cryptanalysis. He listed the primary barriers to brute force: expertise, nonrecurring time and cost, access to tools and technology, replication costs, reusability, and performance. Based on work with an SIMD computer, he proposed a design of an SIMD computer dedicated to brute-force cryptanalysis.

Kedem spent a long time presenting a table, which is in the paper, comparing the experience, cost, and time necessary for brute force when using software, FPGAs, ASICs, and custom chips. Because the talk spent so much time summarizing past research, the audience mostly lost touch with the contributions from this paper.

A Single Instruction Multiple Data (SIMD) machine is made of a large array of small processors. This configuration makes it possible to get close to the theoretical limits of the processor. PixelFlow is an SIMD machine made of many flow units. Each flow unit includes an array of 8,192 processing elements. Using 147,456 PixelFlow SIMD processors, Kedem was able to perform brute-force cryptanalysis of 40-bit RC4 (38,804,210 key checks/second) and the UNIX crypt scheme

(24,576,000 UNIX password checks/second). PixelFlow could try all 40-bit key combinations for a particular RC4 ciphertext in about 7.87 hours.

Because UNC Chapel Hill created the PixelFlow machine for image generation, it does have some limitations when used for cryptanalysis. It has few registers, no dedicated shift unit, no pipelining, and no memory indexing. The lack of memory indexing prevents fast table lookups. Had PixelFlow used memory indexing, Kedem explained, there would have been a 64X speedup for RC4 and a 32X speedup for DES (but the speedups from Kedem's talk are twice that of the figures in the paper). These limitations are specific to PixelFlow, not SIMD machines in general. Kedem then proposed an SIMD design for brute-force cryptanalysis and compared this to FPGA-based machines.

Adi Shamir's TWINKLE project was one buzzword mentioned in this talk. However, an audience participant pointed out that TWINKLE does not take into account that LEDs fade over time.

For information related to brute force cryptanalysis, see <http://theory.lcs.mit.edu/~rivest/bsa-final-report.ps> or contact Kedem at <kedem@cs.duke.edu>.

Antigone: A Flexible Framework for Secure Group Communication

Patrick McDaniel, Atul Prakash, and Peter Honeyman – University of Michigan

Patrick McDaniel introduced Antigone, a flexible framework for defining and implementing secure group policies. To demonstrate the usefulness of Antigone, McDaniel integrated it into the `vic` secure video conferencing system.

Antigone is unique in that it focuses on the following goals:

- Applications can flexibly use a wide range of security policies.
- The system supports a range of threat models.
- It is independent of specific security infrastructures.
- It does not depend on the availability of a specific transport protocol.
- The performance overheads are low.

Taking into account that security policies vary from application to application, Antigone provides a basic set of mechanisms to implement a range of security policies. First, a session-rekeying policy allows sensitivity to certain membership changes including `JOIN`, `LEAVE`, `PROCESS_FAILURE`, and `MEMBER_EJECT`. Second, an application message policy guarantees types of security such as confidentiality, integrity, group authenticity, and sender authenticity. A third policy specifies what kind of membership information other members can obtain. The fourth policy determines under what circumstances Antigone can recover from failures.

One participant asked how to demonstrate that all policies are complete. McDaniel explained that his current work does not have a complete answer. Another participant asked for comparisons between Antigone and other software to implement security policies. McDaniel responded that Antigone currently does not take into account peer groups, voting, negotiating protocols, or ciphers. However, he sees no fundamental reason preventing Antigone from addressing these issues.

In the future, McDaniel hopes to investigate adaptive and role-based policies, implement new mechanisms, benchmark Antigone, and integrate the software with real applications. For more information, see <http://antigone.citi.umich.edu/> or email <pdmcdan@eecs.umich.edu>.

Session: Potpourri

Summaries by Patrick McDaniel

A Secure Station for Networking Monitoring and Control

Vassilis Prevelakis, University of Piraeus

As many system administrators have discovered, managing the network infrastructure for multiple independent communities is a difficult task. Vassilis Prevelakis presented the requirements, design, and experiences with the deployment of a secure network station. The Network Monitoring Station consists of a collection of off-the-shelf hardware and software used to securely manage and monitor remote sites within the Greek University Network (GUnet). The later portion of the presentation discussed the integration and operation of the stations within the target networks.

The target environment presented by Prevelakis consists of independent campus networks that are managed and monitored by a central GUnet network control center. The goal of the architecture is to provide secure, universal access to the network entities within GUnet. This goal is achieved through the deployment of network stations within each remote site.

The initial requirements of the network-monitoring station limited the number of potential solutions. The target networks contained hardware from multiple vendors. Software running on these network components has security services of variable availability and quality. Because of the heterogeneity of networking hardware, no single interface is available. Thus, the design must flexibly support a number of management interfaces, some of which may be unknown at design time.

The hardware constraints were equally daunting. Each station is required to be built using decommissioned personal computers. Moreover, because of reliabil-

ity problems, the use of hard disks was deemed undesirable.

Having presented the target environment and architectural constraints, Vassilis outlined the station architecture and operation. The primary design decision was the identification of the station operating system. Hardware constraints immediately disqualified Windows-like platforms as potential solutions. Next, a number of UNIX platforms were analyzed. Because of the out-of-the-box availability of IPSec and the history of good security-service design, the OpenBSD 2.3 UNIX variant was selected. Because of the lack of a hard drive, Vassilis determined that each station must be booted from an OS image contained on a single floppy disk. Using PICOBSD configurations and `crunchgen` utilities, the OS, system utilities, and station-specific configuration data are compressed onto a single floppy disk. The floppy disk is used to boot a station and may be removed thereafter.

Inter-station communication is primarily based on IPSec tunneling. Where IPSec is unavailable, as in administrative workstations running Windows, `ssh` is also supported. A problem encountered during deployment was the large number of security associations to be configured. This problem is addressed by the automation of the association-generation process from a single station configuration database. However, it is acknowledged that the distribution of new SA information after database modification creates significant administrative overhead.

The network monitoring stations were required to provide facilities for both network monitoring and management. Monitoring facilities allow the `tcpdump` collection of network traffic to be delivered to a logging station over an IPSec tunnel. In monitoring the station itself, the `syslog` data can be delivered similarly.

Management of the remote site is achieved by accessing the station directly or through SNMP interfaces. In those instances where a network object (host, router, bridge) does not support appropriate security services, a station may be used as a proxy to the object's serial interface. In these cases, an object's management interface may be accessed only via the network-management station.

A question from the audience identified a limitation of the existing architecture: the lack of key management services. A change in the IPSec SA database requires the re-creation of boot disks for all stations. This requires the physical involvement of administrators at each site and for each station. Vassilis stated that this problem was to be addressed in the near future, but in practice can be avoided in a large number of cases. It was stated that new stations can be added without affecting the entire network. Thus, at the cost of connectivity, additional stations can be added without requiring all other stations be notified of the change.

Another limitation identified by the audience was the boot disk itself. Because the boot disk has limited capacity, the number of utilities that can be made available is small. The speaker noted that not only are the disks small, but they are an outdated technology. He is currently investigating, among other possibilities, booting over the network and from a CD-ROM.

The Flask Security Architecture: Systems Support for Diverse Security Policies

Ray Spencer, Secure Computing Corporation; Stephen Smalley and Peter Loscocco, National Security Agency; Mike Hibler, Dave Andersen, and Jay Lepreau, University of Utah

Stephen Smalley began his talk by indicating that previous operating-system-level security architectures were deficient in at least one of the following areas: control of propagation of access rights, enforcement of fine-grained access rights, or revocation of existing access rights. The result of a DARPA- and Air Force–funded project, the Flask architecture is intended to address all three of these requirements simultaneously. In addition to meeting these design objectives, Flask is required to have low impact on the performance of user applications and system services.

At its most basic level, the Flask architecture provides support for the flexible definition and enforcement of (potentially dynamic) operating-system-level security policy. Based on the Fluke microkernel operating system, a prototype of the Flask architecture has been developed and benchmarked. This work represents extensions to the authors' previous work on the DTOS architecture.

The Flask architecture defines two subsystems to be integrated into a target operating system. The security server subsystem uses local policy definitions, operation context, and policy-specific code to make security-related decisions. As directed by decisions made by the security manager, the object-manager subsystem enforces policy over the set of objects within the operating system.

A key difficulty of dynamic policy support addressed by the Flask architecture is in providing atomic revocation of previously granted access rights. Without atomic support for revocation, the propagation of the dynamic policies within the system may not be deterministic. This may result in inconsistent policy enforcement.

Flask addresses revocation by requiring that the invalidation of a granted right be a lightweight operation at each object manager. Thus, the security manager may quickly invalidate the right at each object manager that is affected by a particular policy change. Using this approach, Flask

may ensure the atomicity of the change with respect to policy decisions.

The number of policy decisions resulting from even simple user actions may be large. Thus, the performance of the Flask architecture may be limited by the cost of the interactions between the security and object managers. An observation made by the authors is that subsets of these policy decisions are often closely related.

The Flask access vector cache (AVC) is used to limit the amount of communication between the object and security managers. In responding to a policy-decision request, the security manager provides a vector of policy decisions related to and containing a response to the original request. This vector is cached in the AVC, from which the needed policy decision is obtained. Subsequent policy requests that can be serviced by the cached vector can be completed without the involvement of the security manager.

Historically, the performance of operating systems supporting fine-grained security policies has been poor. In the interest of determining the cost of the Flask mechanisms, a prototype was developed and compared with two existing operating systems. Fluke, a capability-based OS from which Flask was derived, outperformed Flask by 5% for a simple `make` task. FreeBSD outperformed Flask by 100% on the same task. It was found that the caching of related policy decisions significantly reduced the IPC costs between managers. The authors were encouraged by these results and indicated that additional optimizations were being considered.

A member of the audience asked for a clarification of the meaning and operation of polyinstantiation within Flask. Used typically as a isolation mechanism, polyinstantiation is the duplication of an object to be used by two or more processes. A cited example of polyinstantiation is the `tmp` directory, where a security domain may wish to protect temporary files from other domains. Smalley continued by describing the mechanisms Flask uses to map operation context to instances of polyinstantiated objects.

A Study in Using Neural Networks for Anomaly and Misuse Detection

Anup K. Ghosh and Aaron Schwartzbard, Reliable Software Technologies

Aaron Schwartzbard enthusiastically presented the results of a study that applies the learning capabilities of neural networks to intrusion detection. He began with a description of the limitations found in existing intrusion-detection approaches. Because of the lack of mechanisms that generalize knowledge of known attacks, detecting new attacks is difficult. Although it's not entirely addressed in this work, Schwartzbard identified the misidentification of normal behavior as attacks (false-positives) as a challenge.

An important aspect of any intrusion-detection system is the fundamental detection approach. In anomaly-detection systems, departures from normal behavior are identified from models of expected activity. Conversely, misuse-detection systems scan activity logs for instances of known aberrant behavior. Thus, depending on the type of approach taken, intrusion-detection systems develop profiles of normal or aberrant behavior on the basis of previously collected event data. The logs used for analysis are typically obtained from traces of user commands, network traffic, or system calls. It has been found that the type of log used has an effect on the performance of the intrusion-detection algorithm.

A limitation of anomaly-detection systems lies in the specification of normal behavior. As user activities change over time, the static nature of the profile may lead to false positives. Moreover, if an attack is used in the generation of the profile, it will thereafter be deemed normal behavior. However, this approach has the advantage that novel attacks may be detected.

Schwartzbard noted that misuse-detection systems typically identify *signatures* of known attacks. The system logs are scanned for occurrences of these signatures, and matches are flagged as attacks. Unfortunately, this approach will only detect attacks that fit signatures obtained from training data.

In attempting to address the limitations of existing approaches, the authors apply a machine-learning approach to intrusion detection. Using training data to develop weights and activations, a back-propagation neural network is developed for each program to be monitored. During subsequent analysis, event data is encoded and fed into the network. The numerical output of the network is then used to identify potential intrusions.

Because detecting attacks on the basis of entire sessions is difficult, logs are typically analyzed using *n-grams* of contiguous events. However, identifying an attack from a single n-gram can be similarly difficult. To combat these problems, trends within collections of n-grams are used to identify intrusion. The size of an n-gram collection and the weights applied to individual n-grams are parameters of the detection algorithm.

Based on the weights and activations derived from training data, the output of a network over some n-gram is a value within the interval (0 .. 1). Thus, the approach does not identify specific behavior, but indicates the amount of anomalous (or normal) behavior within a particular n-gram. When the sum of the weighted output values of n-grams within a collection exceeds a sensitivity threshold, a potential intrusion is flagged.

The authors analyzed the effectiveness of their approach using the 1998 DARPA Intrusion Detection Evaluation program corpus of data. The corpus training data identified both normal and anomalous

behavior within a number of system logs. The training data was used to create a network specific to each program to be analyzed.

The experimental results presented the effectiveness of intrusion detection as a function of the sensitivity of the system. As an algorithm becomes more sensitive, more intrusions are detected (defined as the percentage of real intrusions). Similarly, the rate of false positives (defined as the percentage of misclassified nonintrusions) increases with sensitivity. An ideal system would have, at some sensitivity, perfect detection (100% detection) with zero false positives (0%).

Using test data, anomaly detection using neural networks was able to achieve a high detection rate (77.3%) with very few false positives (2.2%). However, as the sensitivity was further increased, the false-positive rate increased dramatically without significantly affecting the number of intrusions detected. Schwartzbard noted that these results were comparable to existing approaches.

The use of neural networks for misuse detection did not fare as well. High false-positive rates (5%) were observed in tests resulting in even modest detection rates. The authors state that these results are due in large part to the limited amount of intrusion-training data within the DARPA corpus. Future work will attempt to better classify this approach using more substantial intrusion-training data. However, the authors were encouraged by high detection rates found (90%) at sensitivities resulting in relatively low false positives (18.7%).

The long stream of questions indicated the audience's interest in the work. A member of the audience asked for clarification of the "leaky bucket" approach used in analyzing the collections of n-grams. In attempting to detect an intrusion, the value of the previous analysis is multiplied by some value less than or equal to 1 and combined with the value of the current network output. Thus, the weight applied to the value of a single n-gram output decreases (leaks) over subsequent analyses. Schwartzbard noted that the performance of the algorithm may be greatly affected by the multiplier (leakage).

Another question was directed at the fundamental intrusion-detection mechanism, not at the authors' application of neural networks. It was noted that because only n-grams of contiguous events are analyzed, it may be possible for an adversary to shape attacks to be consistent with normal behavior. The speaker noted that this was a known problem, and that current research is investigating approaches that address this limitation.

Session: Security Practicum

Summaries by Kevin Fu

The Design of a Cryptographic Security Architecture

Peter Gutmann, University of Auckland

Peter Gutmann gave a fast-paced talk on how to design a versatile, multiplatform, cryptographic architecture. The implementation works on many platforms ranging from 16-bit microcontrollers to supercomputers and ATMs.

Most security toolkits specify an API, not an architecture. In contrast to the traditional outside-in approach, Gutmann's architecture takes a general cryptographic architecture, then wraps an interface around it.

Gutmann built his architecture based on two concepts: objects encapsulate the architecture functionality, while a security kernel enforces a consistent security policy. Each object has tightly controlled I/O for security reasons. Objects are either action objects (e.g., encrypt, decrypt, sign) or container objects. The containers further decompose into three object types: data containers, key and certificate containers, and security attributes.

Gutmann found that C did not work well for implementing this architecture. The implementation comes in several languages, ranging from C/C++ to Perl and Visual Basic. Gutmann also wrote a formal specification and used the Assertion Definition Language (ADL) to verify his code.

An object can be in one of two states, low or high. In the low state, one can perform all allowed operations on the object. In the high state, one can perform only a limited, safe subset of those operations. An audience member asked whether Gutmann's design prevented an object from selecting from more than two states (i.e., whether it was implemented as something like a single-bit flag). In Gutmann's experience, two states are sufficient. The security kernel supports an infinite number of states, but expecting the user to manage them all is very complex (they would have to track a complex FSM [Finite State Machine] as an object moves from one state to another), and so far there has not been any real need to use more than two.

Everything is implemented in Gutmann's cryptlib library at <http://www.cs.auckland.ac.nz/~pgut001/cryptlib/>.

Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0

Alma Whitten, Carnegie Mellon University; J. D. Tygar, University of California at Berkeley

Alma Whitten raised serious issues about deficiencies in cryptographic user interfaces. After explaining user interfaces (UI), she spoke about the critical results of a usability evaluation of PGP 5.0.

Recognizing that security is fundamentally different from traditional software, Whitten defined usability for security as:

- A user can tell what needs to be done.
- A user can figure out how to do it.
- A user does not make dangerous errors.
- A user does not get annoyed and give up.

Whitten noted that applications such as word processors would focus on the second point. But one cannot assume the second point in security. For instance, in security the undo function cannot reverse the accidental mailing of plaintext.

Whitten explained that "security is not a word processor" because of:

- unmotivated users (security is a secondary goal)
- the barn door problem (the undo problem)
- the abstraction problem
- the software being only as strong as the weakest link
- lack of feedback

Whitten chose PGP 5.0 with the Eudora plug-in on a Macintosh for a case study because, by general consumer-software standards, the interface was reasonably well designed. Her team used two methods to evaluate PGP: laboratory testing (objective, limited, and expensive) and cognitive walkthroughs (subjective, comprehensive, and cheap).

The cognitive walkthrough showed that visual metaphors needed more thought. For instance, the quill head can confuse users. One user mixed up a .sig file with signatures. There is also information overload with too much visual data, such as displaying the whole key ring and metadata. With respect to barn door and feedback problems, we need more protection against irreversible errors such as sending plaintext.

In the laboratory test, users had 90 minutes to perform a series of actions, working in the scenario of a political campaign. The user had to securely coordinate five virtual campaign members. The 12 participants were of a variety of ages between 20 and 55, with educational backgrounds ranging from some college to a doctoral degree, and backgrounds from fine arts to computer technology. Half of the participants were male.

Everyone pretty much could generate key pairs (after a few users misunderstood by generating keys for each campaign member). Twenty-five percent of the users managed to send plaintext instead of ciphertext. Two of these three people realized the mistake. Several users encrypted the message with their own public keys instead of the recipient's key. Nearly everyone fell into this trap. Eventually, after receiving error messages from the virtual campaign members, the participants were able to send encrypted mail correctly. Half of the participants eventually encrypted a message. A fourth of them did it without much help.

By the time the first tests were done, only five people got far enough to decrypt a message. Most could decrypt. However, some mixed up ASCII-armored keys with PGP messages, since the two look the same. The study concluded that PGP 5.0 with Eudora has a nice UI, but competent people in the target group could not handle this. Whitten suggests that to fix these deficiencies one should simplify the UI, minimize what is less important, and add the right kind of help.

Someone suggested that maybe the problem is the PGP model. Can we get a reasonable interface with PGP? Whitten responded that her group looked at where the PGP model did not match users' expectations.

Avi Rubin asked how much documentation and training the test subjects received. Whitten gave each subject printed, bound copies of the Eudora and PGP manuals in addition to a quick tutorial on how to send email with Eudora. In other words, the test subjects had more resources than most users. Moreover, they read the manuals.

Another audience member asked how many users did an out-of-band check to see if the encrypted messages worked. These details are in the paper, but Whitten noted that one technical person noticed the keys were signed and understood that trust in the key had to do with signatures on the key. The majority of users did not understand the trust metric.

Derek Atkins humorously defended himself by saying he designed much of core API, but not UI, for PGP. He asked about problems relating to confusion among various key types. Whitten said that one virtual campaign member had an RSA key and would complain to the test subjects about problems decrypting email. Only one subject figured out that the email had to be encrypted once with RSA and once with DSA.

Greg Rose added an anecdote: USENIX used to run a heavily scripted keysigning. It worked well, but about two-thirds of messages required human interaction.

Another participant asked about the importance of interruptions (e.g., dialog box warnings) to the user about security. Whitten explained that there was no time to look at user tolerance levels. Ideally, one would make the UI sufficiently obvi-

ous to prevent such dangers in the first place.

Questioned about how many of the problems resulted from poor interaction with the Eudora plug-in versus the core PGP interface, Whitten explained that it is hard to distinguish. The UI needs a display to make obvious what is encrypted and what is not. For further information, send email to <alma@cs.cmu.edu>.

Jonah: Experience Implementing PKIX Reference Freeware

Mary Ellen Zurko, John Wray, Iris Associates; Ian Morrison, IBM; Mike Shanzer, Iris Associates; Mike Crane, IBM; Pat Booth, Lotus; Ellen McDermott, IBM; Warren Macek, Iris Associates; Ann Graham, Jim Wade, and Tom Sandlin, IBM

John Wray described the reference implementation of the Internet Engineering Task Force's (IETF) Public Key Infrastructure (PKIX). Wray explained the motivation behind the project. IBM needed a pervasive, open PKI for business; major vendors pledged support for PKIX; and there was a need for reference code to exercise the PKIX specifications.

The team implemented the PKIX specifications as a C++ class library that consists of several RFCs for X.509, Certificate Management Protocols (CMP), Certificate Request Message Format (CRMF), and LDAP V2.

Wray highlighted what the team learned from this experience. What did they do right? They used:

- a single data structure for protocols and persistent data
- C++ destructors to minimize memory leaks
- a single back end, good for promoting code reuse
- an unfamiliar platform (NT) for development

However, Wray also noted what they did wrong:

- no proper error architecture (but Wray has never seen a good one)
- interoperability testing too late
- sloppiness with respect to case sensitivity (NT is case-insensitive, but UNIX is case-sensitive)
- STL (Standard Template Library) problems

Asked how easy is it to use the CDSA architecture, Wray replied that indeed CDSA 1.2 presented difficulties. Questioned about certificate revocation, he replied that the implementation publishes certificate revocation lists.

For more information, see <http://web.mit.edu/pfl/> or read the archives of <imc-pfl@imc.org> on <http://www.imc.org/imc-pfl/>.

Session: Access Control

Summaries by Matt Heavner and Ping Liu

Scalable Access Control for Distributed Object Systems

Daniel F. Sterne, Gregg W. Tally, C. Durward McDonnell, David L. Sherman, David L. Sames, and Pierre X. Pasturel, Network Associates, Inc.; E. John Sebes, Kroll-O'Gara Information Security Group

Gregg Tally presented ongoing work to extend CORBA with a desire for fine-grained access (per object, per operation) in order to facilitate the widespread use of distributed object-oriented systems. DTE (Domain & Type Enforcement) is a set of access-control mechanisms for UNIX kernels that was extended to distributed-object systems and applied to CORBA as OO-DTE, a plug-in (for the ORB) that uses SSL (for domain checking). OO-DTE uses a distributed policy scheme (with a master policy server using CORBA connections to distribute policy to local policy servers). The talk featured details regarding an example application to manage books (book check in/out and query) in a library. The example is presented in the conference proceedings paper, but was extended a bit in the talk.

Tally briefly compared OO-DTE with CORBASec (a spec released in 1996): OO-DTE permits the use of wild-card rules to facilitate the assignment of types to methods. CORBASec requires the enumeration of rights for each interface, without inheritance, making specifying new interfaces more tedious.

One feature of the DTE work was DTEL, a high-level compilable language for implementing security policy. OO-DTE includes DTEL++, which provides constructs for assigning types to methods.

Tally summed up the results of the work as follows: OO-DTE is currently implemented as a plug-in to ORBIX and Visibroker; it is scalable and flexible (allowing both coarse and fine granularity); with DTEL++ there is a high level of ease of administration (policy creation through a set of general rules and exceptions, using inheritance from base interfaces to derived interfaces), so large numbers of objects can be "labeled" with just a few DTEL++ policy statements. Benchmarking reveals that SSL and Intercept take more overhead than the OO-DTE component.

The OO-DTE and DTEL++ implementation code is available from Tally at <gtally@nai.com>.

Several people questioned the interoperability between different ORBs, since the solution depends on the object keys, which unfortunately are different from vendor to vendor; the answer is that there are no good solutions yet. Peter Honeyman suggested using references such as LDAP to solve the problem, but the feasibility needs to be investigated. Another audience member asked if OMG plans to standardize the object key. The answer is that it's fundamentally not supported (big laugh).

Certificate-based Access Control for Widely Distributed Resources

Mary Thompson, William Johnston, Srilekha Mudumbai, Gary Hoo, Keith Jackson, and Abdelilah Essiari, Lawrence Berkeley National Laboratory

Mary Thompson presented the current state of the Akenti system for certificate-based access control. The motivation for Akenti is a distributed computer and collaborative-use system, shared between organizationally and geographically distributed facilities and users. The goals of the project are to implement policy-statement-based access, to provide multiple-stakeholder control of a single resource structure, and to use a public key system. The emphasis for the system is on usability. Currently, the PKI used consists of X.509 certificates, SSL, and digital signatures. (PKI deployment is more of a future issue, not so important on the small scale for which Akenti is currently being designed.) Akenti is implemented as several library routines used in conjunction with Apache with minimal local policy files (whom to trust, and certificate location).

A major feature of Akenti is the GUI for designing access policy, including the ability for users to check access lists in order to see what portion of the infrastructure may be blocking access (e.g., if a stakeholder has written a "bad" rule); this may be changed so only stakeholders can view access lists in larger implementations.

Some vulnerabilities in the Akenti system are due to the distributed-certificate nature of the system: a certain level of trust on possibly insecure remote machines is required, and network outages can be disastrous. A problem in the rule-set implementation is that independent stakeholders can create mutually exclusive access policies and unintentionally lock out users. A performance view of Akenti shows the following features: it has fairly high granularity; 80% of Akenti time is tied up in fetching certificates; and search and failure takes more time than a successful lookup. The current status of the project is that Akenti/Apache is in use at LBNL and Sandia for control of Akenti code distribution and access control for notebooks.

Currently Akenti runs on Linux and Solaris. Future modifications to the Akenti project include the use of XML certificates, a standalone implementation (unlinked from Apache), expanded use conditionals, and possible implementation of bandwidth-control policies.

There were four questions at the end of the presentation. Q: Is there a problem with certificate storage for use by both Netscape & IE users? A: There used to be, but it seems to be working now. Q: Use of Netscape certificate generation ("click hell"). (The question was a plug for the "PK no I" WIP by Honeyman.) Q: Is there a problem with a "spoof" attack to get around "not" rules? (Meaning: if there is a rule that "Joan from Sun can't use the Coke machine," then Joan can subvert this by logging in to request access to the Coke machine through a non-Sun account.) A: This is a possible problem: the full range of access-control implementation is still a work in progress. Q: Is there a problem with revocation of rights? A: The revocation of rights is currently to be implemented at "host institutions" and propagated through the Akenti system.

Digital-Ticket-Controlled Digital Ticket Circulation

Ko Fujimura, Hiroshi Kuno, Masayuki Terada, Kazuo Matsuyama, Yasunao Mizuno, and Jun Sekine, NTT Information Sharing Platform Laboratories

The last paper in this session is about a digital-ticket-circulation and corresponding trust-management scheme. The requirements the authors are trying to address are:

- flexibility to satisfy different business purposes
- flexible and automatic management
- simple verification

The author proposed an "Onion Ticket Accumulation Model," with the user's identification information as the onion core and the user's rights forming the outside layers. A related trust-management scheme was also proposed, aiming to provide some means of ticket verification regardless of the circulation route. The approach is quite ad hoc: digital signature, hash value, plus the ticket type identifier information.

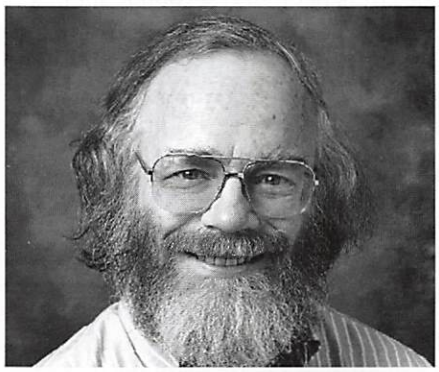
The authors hope to be in IETF discussions soon and to be published soon. Information will be at <http://info.isl.ntt.co.jp/flexticket/>.

INVITED TALKS

Cryptography and the Internet: Networks and Security and Why the Two Don't Get Along

Steven M. Bellovin

Summary by Matt Heavner



Steven Bellovin, filling in for Peter Neumann (who gave the keynote address), gave a talk originally presented at CRYPTO '98. The talk fitted in well with the Security Symposium.

The rhetorical question that started the talk was: "Why is cryptography coming to the Internet now?" The answer is that the driving force of money has been applied in the form of e-commerce. As businesses attempt to extract money from consumers via the Internet, there is a public perception of insecurity and also a public perception that cryptography is an answer. Therefore the "solution" of cryptography is getting stapled onto the Internet. Current cryptography use on the Internet is primarily for email (PGP and S/MIME) and the browser use of SSL. Up-and-coming cryptography includes IPSec and SET.

The current use of cryptography in email (PGP, S/MIME) suffers from a scaling problem and the lack of a true PKI necessary for widespread (pervasive) use. Although SSL is not limited to HTML, that is the overwhelming use of SSL. Also, although SSL is meant to provide a PKI solution, such a thing does not actually exist, because users don't know what certificates are or what to do with them!

IPSec is a more general cryptographic solution than either of the two above (since it is implemented at the IP layer and therefore does not require modification of each application). The initial use of IPSec will be for VPNs (where no general PKI solution is required). With the possible implementation of IPSec in upcoming Microsoft and Freenix OSes, the widespread deployment should be a reality within three years (but does this mean that it will be widely used?). One of the biggest problems with IPSec is reality (the use of nonsecure computers for secure work, e.g., a former CIA director's recently revealed work habits).

SET is an alternative to the widespread use of credit card (CC) numbers on the Internet. SET may replace CC numbers on servers. It is a multiparty protocol, in that it involves the consumer, the bank, and the vendor. The SET system will not be widespread without a monetary incentive from the CC companies (such as lower rates for merchants). However, the CC industry may be interested enough in the implementation of SET that it could happen relatively soon and become widespread.

The next phase of the discussion concerned what is still missing for widespread use of cryptography on the Internet. First, the speed of public-key operations creates a bottleneck at the server – modern CPUs can handle large public keys, but servers get bogged down with multiple simultaneous requests.

Second, secure routing on the Internet is a difficult problem, for two reasons: the topology of the network (generally many hops between hosts) and the lack of well-defined secure and/or synchronized time. The time problem is difficult because the lack of a trustable time standard makes any time-stamping of validity periods impossible. One possible solution to the topology problem may be a "chain of digital signatures" between trusted pairwise connections along the entire path between hosts. However, the backbone routers are already pushed to their capacity, and this would require many digital-signature verifications. A problem exists in determining the currently correct route for traffic – the time for which a given route is correct is small – therefore, a route that may very recently have been correct may no longer be correct. Another similar problem is secure multicasting.

One problem found in trust-management issues is the conflict between machine and human understanding of trust management. One example is a certificate presented by interactive.wsj.com which was issued for www.wsj.com. Obviously, this is not the same as the difference between nasa.gov and nasa.com, yet Web browsers will have trouble with the mismatch in names.

Another problem in getting cryptography integrated into the Internet is one of "cryptography versus cryptographic engineering." A cryptographic paper may propose a system, and setting up a "standard" implementation (made of key lengths, etc.) is the next step. Then the software implementation is a completely different problem. There are several requirements for Internet cryptography, exacerbating the theory-versus-engineering problem. Fine granularity is desired, but this simplifies the types of attacks to which the crypto system is vulnerable.

Returning to the DNS/routing issues, Bellovin brought up the specific issue that the DNS TTL field changes constantly, so the DNS record cannot be signed! Also, secure DNS needs to provide dynamic updates, as well as negative answers to queries, within a reasonable period of time.

Several examples of the conflict between cryptographic requirements and "real world" network requirements can be found in IPSec. IPSec hides everything,

so traffic analysis by network engineers is no longer possible. Network-address translators cannot deal with IPSec (if they can't get at IP and port number, it is not possible for them to translate). Adjusting the windowing of network traffic for transmission over satellite link latency, as well as "tinkering" with the transmission over wireless nets, cannot be done with IPSec in use.

One final difficulty with cryptography and the Internet is protocol verification. Designing a cryptographic protocol is hard enough, but reality can be even worse! The common attitude of "shoot the engineers and ship the product" to get products to market, and the problem of "late science" (a flaw in cryptographic implementation found years after product distribution) are additional headaches associated with the widespread use of cryptography on the Internet.

In conclusion, Bellovin pointed out that no more than 15% of CERT advisories in 1998 to mid-August could be solved/avoided with widespread deployment of cryptographic solutions. (For example, many of the problems are still simple buffer overflows.) One last problem is that a lot of bad crypto is out there (e.g., a self-extracting crypto message system was sold as a means of eliminating the need for secure exchange of secret keys!).

The viewgraphs for Bellovin's talks are online at <http://www.research.att.com/~smb/talks/inet-crypto.ps>.

The main subject discussed in the question period after the talk was Public Key Infrastructure. The points made were that PKI is a top-down system, whereas the real world is not top-down. So it comes down to anonymous trust in the real world. One major problem is the lack of user knowledge – as an example, Netscape ships browsers signed with expired keys.

Rik Farrow asked about the problems of interoperability among current IPSec implementations. Bellovin answered that it works well at the IP level; it is only so-so at the certificate level; and it is bad if different certificate authorities are used. Hopefully this will improve on a six- to 12-month time scale.

US Crypto Policy: Explaining the Inexplicable

Susan Landau, Sun Microsystems Laboratories

Summary by Jim Simpson

A very well-informed and eloquent person, Susan Landau is a senior staff engineer at Sun Microsystems Laboratories. She also co-wrote the multi–award-winning book *Privacy on the Line: The Politics of Wiretapping and Encryption* (which Avi Rubin, who introduced Landau, highly recommends to anyone in the field of computer security and to those looking to better their comprehension of the field). Her informative talk to a crowded room, punctuated with witty comments, explained the context of the current US crypto policy.

In the period between 1764 and the early 1800s, the founding fathers of the US used cryptography when communicating about love, life, and – most important – politics; they understood the importance of maintaining integrity. In 1999, the government takes a contrasting stance and claims that the same idea of protecting communication will nullify its ability to investigate acts of terrorism and protect citizens. In 1992 the FBI indicated that by 1995 strong crypto would make 40% of wiretaps ineffective; in 1995, out of all wiretaps, only one or two were not understandable. Clearly, their concerns remained unjustified.

Landau presented the Fourth Amendment to frame the issue and explained it thus: even though the government can obtain the right to search, it does not have the right to find. She indicated that this difference is very important, and further explained that the government is actually more concerned with wiretaps than with email. In a case in which the Supreme Court did not agree that wiretapping someone was a violation of the Fourth Amendment, a dissenting judge suggested that wiretapping is really no different from the writs of assistance the colonists rejected. Finally, Landau mentioned the moment in the Watergate hearings where Senator Talmadge reiterated the importance of the concept of privacy, in response to another senator's deprecating comment about how circumstances had changed since the Fourth Amendment was written.

Many Americans do not realize privacy is not necessarily guaranteed by the Constitution, even though there are amendments that allude to privacy. Most rulings on privacy come from case law. Some cases have ruled in favor of privacy while others have not, indicating that privacy is not always clearly defined. This is one of the reasons why the US crypto policy seems to be inexplicable: cryptography helps guarantee privacy in communications.

In 1934, Congress passed the Federal Communications Act, which in effect said the government may not tap and divulge wired communication. A court case prior to the law ruled in favor of those doing the wiretaps, while a case afterward ruled in favor of those being wiretapped. There had actually been a similar law at the time of that early case in 1928, but the lawyer did not think to use it. When Germany invaded Russia in 1939, J. Edgar Hoover got permission for the FBI to intercept – but not divulge – communications by wiretap, and this activity continued throughout the war. By the time Truman was in office, the last sentence of the executive order allowing these wiretaps, which limited targets of wiretapping to foreigners, mysteriously disappeared. The FBI worked very hard to keep these wiretaps undiscovered over the next 30 years, by marking files as confidential and using information from "confidential

sources" when in court. Whenever Congress tried to investigate the possibility of wiretaps, it was silenced.

In 1967, the case of *Katz* v. *US* led to the Supreme Court ruling that no bugs be installed without search warrants, effectively throwing out wiretaps and bugs in court cases. However, because of mounting social tensions involving organized crime, law-enforcement officials pushed for the use of wiretaps, which soon led to the creation of Title III, which did indeed allow wiretaps. Finally, in the late '70s, another law approved the use of wiretaps in foreign-intelligence surveillance.

Title III specifically covers the use of wiretapping. It states that wiretaps should be used only as a close-to-final resort; the use of a wiretap requires a search warrant, and they can be used only to investigate certain crimes. The cost of a wiretap is up to $58,000 a year, so only around a thousand of them are installed each year. The second law, FISA, made provisions for the use of wiretaps for foreign intelligence only; Americans are not to be targeted. Around 500 are installed annually, and information about them is hard to come by. Organized crime and drug trafficking cases make up about 75% of all wiretaps, which conflicts with the FBI's argument that it uses wiretaps to go after terrorists and kidnappers. Only recently have Title III wiretaps included terrorist activities. Kidnapping is the strongest argument against crypto, as crypto is often presented as being the obstacle that prevents an authority from reuniting parent and child; it turns out that only two or three cases out of several hundred kidnappings actually make use of wiretapping.

Landau feels it is a poor idea to base public policy on two or three cases a year. She pointed out how ineffective wiretapping can be in the case of kidnapping: if you do not know who kidnapped the person, how do you wiretap them? Certainly, if you have the approval of the victim's family to listen in on any incoming ransom call, crypto is not going to stand in the way; if the family can understand the call, law officials can, too. At this point, the room exploded in laughter.

It turns out the US government has no problems with the use of crypto for authenticity and integrity. There are economic arguments for the use of crypto as well. The government's response has been to give control of export to the Department of Commerce instead of the Department of State, and DES is limited to 56 bits. Current policy includes Clipper in 1993, CALEA in 1994, and the AES competition. CALEA essentially states that switching networks have to be built so that they are wiretap-accessible, otherwise the telco would face a $10,000 fine per day for every wiretap it was not able to do. Between 1994 and 1998, standards were to be drawn up for how many simultaneous wiretaps were to be possible, how much carrier capacity, and so on. The Department of Justice decided the FBI would be the best agency to do this. The numbers they came up with in 1995 were for 30,000 simultaneous wiretaps in the US; between Title III and FISA, we only do 1,500, for maybe around 1,000 simultaneous wiretaps. The telcos objected, so in 1996 the FBI came back with a different number: 60,000.

The policy arm of the US government is pushing key escrow. Six years of effort have yet to lead to any sort of key-sharing agreement. As a result of key escrow, a tremendous delay in the deployment of other cryptography has taken place. Devices using key escrow have not done very well in the public sector; out of 15,000 secure phones manufactured by AT&T, 9,000 went to the FBI. Finally, an internal memo written by Bill Reich, who works for the Secretary of Export Administration in the Department of Commerce, summed up the sentiments of those who pushed key escrow: they themselves did not like to use it, since it took longer to initialize.

Landau discussed how the SAFE bill, at one point good because it was to relax export control, was gutted and changed by the Armed Services Intelligence Committee to the point where it would make it a crime not to decrypt when ordered to by a court. Following that, she touched on FIDNET, the Federal Intrusion Detection Network, to discover when there are problems with the network information infrastructure. This would monitor the network information infrastructure; Landau wholeheartedly agrees we should secure the nation's network information infrastructure, but wonders why the government is making it so difficult to do so. Once again, laughter and clapping echoed throughout the room.

Landau is often asked for her opinion of what the future holds here. She feels there is a race going on. This race involves what the NSA and/or the FBI are trying to get away with, and what happens in Europe. If enough European competition starts taking away US business, then Congress may pass a certain set of laws. She also feels the FBI has a much more polarized view than the NSA; the NSA knows the game is lost as far as crypto goes, and it has a vested interest in the security of the US industry. A weak US computer industry makes the job of the NSA that much more difficult. Export controls that hamper the US computer industry are problematic precisely for that reason. She also feels that the currently admired practice of open source may change or influence the policy. If source code becomes public information, it will be harder to enforce export control, since the reason to do so will become less relevant.

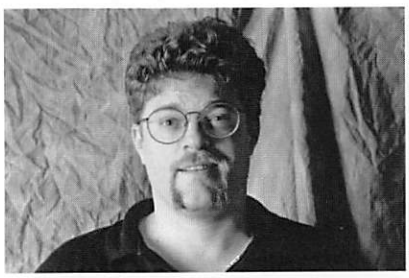
She ended her talk with a poignant story that took place shortly after CALEA passed. The FBI invited law enforcement from around the world to learn how to install and use similar wiretapping capa-

bilities of digital switching technology. One of the police forces invited belonged to Hong Kong, which is now under Chinese jurisdiction; every year the State Department lists human-rights abuses from around the globe, and China is consistently at the top of this list. She ended with twin questions, "What are the technologies we're exporting, and what are the values we're exporting?"

The Burglar Alarm Builders Toolbox

Marcus Ranum, Network Flight Recorder

Summary by Rik Farrow



Marcus Ranum shared some of his experience, while revealing yet another bit of job history, in this presentation. Ranum could not share code examples, because most of his stuff turned out not to be portable to Linux. But there were more than enough ideas to keep you busy for a while, and perhaps even trip up an almost successful attacker.

He targeted misuse detection, that is, looking for activities that should not occur. For example, his cats are not authorized to portscan his servers, and if they do, something must be amiss. You need to know how your network is actually built to do this; at home, the three large cats are not authorized to open the doors, and when that happens, an alarm will be set off (note that Marcus's cats are all MSCE, so you never know what will happen). You can apply this by setting up misuse detection to detect misuse of site policy. He remarked that this is almost like a security assertion.

Another approach is to watch for second-order effects. Second-order effects are things that might happen after a successful, but not yet detected, break-in. These things include adding a new user account, setting the execute or setuid bit, adding a new service, or making changes to your Web pages. You do this by leveraging local knowledge and knowledge of commonly used attacker tricks. Ranum remarked that he had worked for a while installing burglar alarms for his father's company, and that the second-order alarms were what caught the slicker thieves: switches set under a fake jewelry box or on the door of the gun safe.

If someone avoids all your burglar alarms, you know it's an insider job.

Ranum listed the advantages of watching for the second-order effects: detecting previously unknown attacks, cheap to do, and will rarely set off false alarms if done correctly. The disadvantages include having to understand your network and how things should work, and that this approach is policy-directed (you may not have policy).

Marcus had a long list of suggestions. For example, instead of just patching that buffer-overflow attack, have a failed attack generate a `syslog` message. Use packet filtering on your firewall-protected Web server. Then if anything unexpected reaches the packet filter, you will know that something is wrong. You can also do this with a sniffer, or even `tcpdump` with a simple set of rules that write to a log file; if it gets written to, something has happened. Another example: block stuff from the inside to the outside (IRC from your Web server as an example, a whole list of things that your Web server shouldn't be doing). There are many tools to help with this: hardware sniffers, in-kernel packet filters, applications (Argus, NFR), `tcpwrapper` logs.

Use safe-Finger to reverse finger someone who touches a port. There is the lame example used with `tcpd`, and he also suggested touch files for slow scan detection.

Ranum suggested trapping certain actions by replacing the top half of `exec` system call (or `connect`, `accept`, `chmod`, `exec` system calls). This suggestion sparked some audience participation, as someone suggested that MEMCO (seller of SEOS) had patented the idea of replacing the system call jump table. No one knew for sure, but even MEMCO's own PR mentions that this technique was taken from IBM's OS/360. A quick search uncovered a Linux module for installing trojans in the jump call table (<http://www.rootshell.com/search.fcg>, look for <heroin.c>). Gaspar Carson mentioned debugging tools, and Solaris includes an interface for intercepting system calls.

Other suggestions included whacking your shell when it gets called with `-c`, preventing a second `chroot()`, disabling `fchroot()` (never used, but there for completeness only), trojaning commonly used commands and training yourself not to use them (`ls`, `chown`, `chmod`, etc.), replacing the NIT or BPF driver with something that triggers an alarm. A more humorous idea was to create a program called `watchdog` that does nothing (but creates paranoia).

Along the lines of strange things to leave lying around, Ranum suggested installing things that look like trojans, such as BackOfficer Friendly, redirectors that reflect scans back to their source, and the `phat_warez.zip` file (a couple of gigabytes of zeroes compressed into something quite small).

There was lots more; you can find the complete presentation on Ranum's Web site: <http://www.clark.net/mjr/pubs/index.html>.

ActiveX Insecurities

Richard Smith, Phar Lap Software

Summary by Rik Farrow

Richard Smith is not that well known for his day job as president of Phar Lap Software, a maker of embedded and real-time development tools for x86. But his hobby has helped make him famous. Or, as Smith says, "If Microsoft can build it, he can break it." You might also have heard of Smith as the person who discovered David L. Smith's name found in the Word document that was part of the Melissa virus.

ActiveX controls appeared as VB (Visual Basic) controls on the Web about three years ago. Controls are largely what the X toolkit world called widgets (push buttons, dialog boxes, sliders, etc.), but they can be much more than that. Microsoft, as a predominant vendor, wants to move as many of its technologies as possible onto the Web, for more control. From the users' point of view, it is less concrete why they might want ActiveX on their systems.

ActiveX is DLL (dynamic link library) binary code, and it can be called up and scripted from a Web page. Right from the start, this is a pretty scary thing. HTML is a formatting language, Java and JavaScript are programming languages, but Smith points out that Microsoft is taking a real security risk with ActiveX. Even with digital signatures, ActiveX is a lot scarier than people think. Smith knows of five or six people, including Georgie, a Bulgarian security researcher, who have time on their hands to look at these problems.

What is even more interesting is that you can send HTML via email and expect to have Internet Explorer (IE) interpret it for you. Outlook Express, the mail client shipped with Windows 98 and also as part of Microsoft Office, will automatically invoke IE, which can in turn invoke ActiveX controls. HTML <object> tags contain JavaScript, which can invoke the controls. In his first example, just by reading an email message (from Georgie), there is now a file in his startup folder that invokes `command.com` (and could have invoked delete tree, the Microsoft equivalent to `rm -rf`).

At least five to ten million people run IE5. By default under Outlook, the viewer for messages is IE5. So when you hear about browser exploits, think email also. This means you can direct an attack at particular persons, as long as they use IE5. As far as Smith knows, there is no way to disable HTML interpretation, but you can improve your security by disabling scripting.

Actually, Microsoft did include a method for assuring that only certain controls, marked as "safe," could be invoked from scripts. What's happening is that ActiveX controls can be marked safe for scripting when they are not. If they are marked safe for scripting, they can be used in Web pages (or email). Windows 98 comes with over 900 controls, and thousands more may be added by the vendor and by installing software packages.

Smith and his friends have uncovered about ten controls that are marked safe but are not. His favorite example is the Launch ActiveX control that has been shipped with five million HP Pavilion PCs. With this control, email or Web pages can be used to start any application on the targeted system. Compaq installed a slightly less dangerous control, one that can overwrite any file (`autoexec.bat`, for example). There is also an ActiveX control in IE5 that can steal your PGP secret key.

Internet Explorer does permit you to disable scripting, which will foil these attacks. Of course, you could just use Netscape and not have this problem. But there is another technique that can be used even if you disable scripting. Smith called this the bullying technique. He used the Compaq control as an example. A dialog box pops up, saying that the control will be used only if the user trusts Compaq (based on the authenticode signature). Smith enters "No," and immediately the dialog box pops up again with the same question. Eventually the user is likely to give up and say yes.

Smith mentioned that the problems do not appear as bad with IE4 and Windows 95.

You can view the interfaces to ActiveX controls with OLEview. This is not a standard part of a Windows or NT distribution, but does come with Visual Studio and other developer tools. What would be nicer would be something that would print the IDL (Interface Definition Language) as text, so that UNIX tools could be used to look for interesting interfaces (those named Launch, Write, IObjectSafety, etc.).

Smith provided a URL for pages that will check your system for dangerous controls: <http://www.tiac.net/users/smiths/acctroj/index.htm>. I visited this page (using Netscape and UNIX), but you are really supposed to use it with IE5 (and I think we can trust Mr. Smith not to do anything bad).

There were some questions at the end of the talk. Someone asked what policy Smith would suggest in regard to various scripting languages. Smith said that he feels pretty good about JavaScript and the controls on Java, but doesn't really see much use of ActiveX on the Internet and suggests disabling it. Another person asked about blocking ActiveX at firewalls. Smith responded that this does not work well.

Finally, a person asked plaintively, "Do I tell people it's safe to read email?" Smith responded, "In theory, no, ever since HTML email appeared. Never open up attachments, and get your security settings right in Outlook Express." Sigh.

Apples, Oranges, and the Public Key Infrastructure (PKI)

Paul C. Van Oorschot, Entrust Technologies

Summary by Michael J. Covington

Paul Van Oorschot is a vice president and chief scientist with Entrust Technologies, a spin-off from the Secure Networks division at Nortel. With an extensive background in mathematics, cryptography, and work in the industrial sector, Van Oorschot has been exposed to many of the forces that are motivating the drive for secure computing in today's business marketplace. He discussed the challenges involved in building a flexible, secure, and standardized public key infrastructure (PKI). From certificate creation to contract signing, he discussed the details of this complicated, and still unstandardized, transaction-based process.

Van Oorschot opened his talk by promoting PKIs and by providing a brief history detailing the evolution of certificate-based technologies. As he discussed the details of certificates and the many popular protocols that rely on them, it became apparent that certificate technology is indeed becoming ubiquitous. Unfortunately, the current state of certificate deployment seems to be on a course for disaster. Statistics show that over 150 million browsers have been distributed with the capability to "understand certificates," yet the standards by which these objects are created, destroyed, and maintained have yet to be established.

In addition to lacking a supportive infrastructure for public-key technologies, there remains widespread misunderstandings about PKIs in general. Paul detailed a number of these, from differing deployment markets to complex theoretical and implementation details. His point was clear: there are different markets and different solutions available. There are also various service approaches that affect design. The key (no pun intended) to proper PKI design, however, lies in a thorough understanding of the requirements and well-defined standards.

The most interesting segment of Van Oorschot's talk was his detailed discussion of the fundamental components that make up a PKI and the features that should be incorporated into each. A common misunderstanding is that a certification authority (CA) alone can serve as a PKI. Van Oorschot insists that this is not the case – the CA is but one small piece in a large puzzle. In addition to certificate issuance, there need to be mechanisms in place to provide services such as key generation, key updating, key expiry, and even certificate validation.

Building upon his concept of the all-in-one PKI, Van Oorschot proceeded to present an argument for a single security infrastructure – one in which multiple applications, running on a diverse set of operating systems, could operate seamlessly. Such a system, when placed in the appropriate computing environments, could be leveraged across all enterprise applications and would avoid a great deal of the interoperable complexity and duplication of effort that would arise from multiple systems.

Van Oorschot argued that a "Managed PKI" involves a complete integration of several components into a single, trusted system. With a PKI sitting at the heart of secure e-transactions, it is important that the infrastructure address cross-application security and also provide the key and certificate-management abilities that make the system easy to understand and operate.

Audience discussion that followed the presentation addressed an issue related to actual PKI implementations. Various users in the group were interested in Van Oorschot's comments on PKIs and "mapping services" that distinguish between common names or IDs and internal system identifiers. Although the discussion focused on a particular implementation, concerns regarding widespread usage of public-key technologies were expressed. Clearly, as these technologies are deployed, the ability to look up users and their associated keys accurately will be yet another component of this already rich public-key infrastructure. We can only hope that the standards defining these technologies are introduced soon.

WORKS IN PROGRESS

Summaries by Jim Simpson

USENIX Student Benefits

Peter Honeyman, University of Michigan

As USENIX's elected secretary, Peter Honeyman focuses on academic relationships and scholastic support. USENIX has 7,000 members, grows by about 12% a year, and requires about $5 million a year to run. $1 million goes to good works, which comprise support for directed projects, and – most important to Honeyman – direct support for students in the form of student stipends to attend conferences. Another program, the USENIX Scholars program, spends a third of a million dollars a year supporting graduate-student research in distributed systems. It is easy to write proposals to fund students and their research. Check out <http://www.usenix.org/students>.

An Update on AES Selection

Elaine Barker, NIST

DES has been in use since 1977; AES will replace it in the near future. Twenty-one algorithms were originally submitted over a two-year period, and 15 were analyzed. Of those 15, five were chosen as Round 2 candidates. AES3 will take place in April 2000, Round 2 will end in May, and the winners will be announced in summer 2000. Promulgation should happen in 2001. More information can be found at <http://www.nist.gov/aes>.

Distributed Firewalls

Steve Bellovin, AT&T Labs

Today's firewalls do not meet today's needs. However, firewalls are still very necessary; they are single points of control that block harmful protocols and act as shields for buggy implementations. Bellovin has come up with the concept of a distributed firewall, where control is centralized but doesn't rely on topology, so there is no single point of failure. A system manager uses a high-level language to describe endpoints and specify the security policy. A compiler translates the policy into filter rules that are distributed to all servers. Filtering is done at IPSec layer, so identity matters, not topology. *[Editor's note: See Bellovin's article, " Distributed Firewalls," on page 39 in this issue.]*

T-Class SOBER Stream Cipher

Greg Rose, QUALCOMM

This talk turned out a bit different from what was originally planned. Greg announced a new stream cipher called SOBER during the last Security Symposium, and since then no fundamental weaknesses have been found. At this WIP Greg had intended to announce an enhanced, more efficient version. However, it turns out that in the planned "new SOBER" the stuttering used to protect the nonlinear function is broken; it throws away the strength of the cipher. The people who know more about the cipher believe the original stuttering and cipher are strong. They plan on using the original stuttering with their new optimizations.

Security Risk Management

Andrew Kotulic, York College of Pennsylvania

There are no theoretically based process models in this field, so this is a program/policy to figure out how to organize SRM. The conceptual model analyzes which factors security practitioners have control over and which ones they do not. The industry is not very responsive to this research, but the project is pressing forward by routing around chief security officers in companies and speaking with their executives directly.

PK No I

Peter Honeyman, University of Michigan

Spurred by time constraints, Honeyman amusingly went through his talk backwards. The project is trying to provide access-controlled Web space, but it cannot wait for Kerberos to become integrated in browser space. They are looking at solutions using ApacheSSL, but issuing client certificates is a problem – where do they go? Thankfully, IE uses CAPI, and there is a PKCS#11 plug-in for Netscape to take care of it with that browser. There is currently an issue with IE on the Mac when using IE: no CAPI. They modified an MIT solution so it uses one-day lifetime certificates (junk keys), not useful for anything but authentication. It is working, leveraging off their Kerberos infrastructure. Slides from the talks are at <http://www.citi.umich.edu/u/honey/talks/pknoiwip/sld001.htm>.

Intermediate Protocol Enforcement

Craig Metz, University of Virginia

There are a lot of broken systems, and everyone wants a "drop box" that will make problems go away. In theory we could build a box to do that, since protocols are well specified. Metz is in the process of trying to do this right and anticipates finding more problems than anticipated. The code currently works on IPv4, working on TCP segmentation and reassembly. In current form it seems to work as proposed: things that do not follow protocol will not go through.

The History and Future of Computer Attack Taxonomies

Daniel Lough, Virginia Tech

This is an attempt to classify attacks into different categories. Security failures occur in three areas: protocol design, protocol implementation, and configuration by user. Several different kinds of attacks span these areas. A rise in router attacks in the future is anticipated. Lough will be taking the past attack work by Howard and Bishop and seeing if there are similarities with attacks of the present day. Wireless networks of the future may constitute an even larger problem.

Trust Management and Network Layer Security

Matt Blaze, AT&T Labs

Keynote is a systematic way to answer questions characterized by dangerous actions in distributed systems. This is done by representing and specifying policies, credentials, and relationships among principals. Current work involves an IPSec trust-management architecture; this allows control over which packet filter is installed when a security association is created. Keynote is freely available. More information can be found at <http://www.cis.upenn.edu/~keynote>.

Sun Enterprise Network Security Service (Bruce)

Alec Muffett, Sun Microsystems UK

One of the shortcomings of vulnerability scanners is that they do not scale to WAN size. Bruce is a system of distributed daemons that are linked into a hierarchy that distributes and executes security-checking code. Information is centrally retrieved, collated, and then viewed by a Web browser. They are working on bug fixes, functionality enhancement, and security refinements. *[Editor's note: See Muffett's article, "SENSS Bruce: Developing a Tool to Aid Intranet Security," on page 35 in this issue.]*

Internet Mapping

Bill Cheswick, Bell Labs

For the last year, Bill Cheswick has been collecting Internet maps. He does this by running traceroutes from a machine at Bell Labs to 90,000 destinations, and then saving the data. Hal Burch at CMU helped with the layout algorithm to generate maps from the data. An interesting use is mapping an intranet to find holes in its perimeter and report back; this program can do it in an hour or so, whereas a high-caliber commercial application takes a month. The code is not available, but they are happy to scan your network. They may turn this project into a consulting tool for Lucent. More information can be found at <http://www.cs.bell-labs.com/who/ches/map>.

Secure Remote Access to an Internal Webserver (Absent)

Avi Rubin, AT&T Labs

Absent is a solution allowing an absent individual to access a secure site. This works with a Web browser using a one-time password over an SSL connection that encrypts the channel and allows secure access to Web servers behind the firewall by rewriting URLs. It is fully functional, and the paper is available from <http://www.research.att.com/projects/absent>.

Cool Smartcard Hacks

Peter Honeyman, University of Michigan

Some of this was previously published in the *Proceedings of the USENIX Workshop on Smartcard Technology* last May. Peter went over a Kerberos client involving a smartcard, a file system that integrates with a smartcard, secure booting, Javacard Web servers, and PalmPilot hacks. More information on CITI's work with smartcards is available at <http://www.citi.umich.edu/projects/sinciti/smartcard>.

GSM A5/2 Algorithms Revealed

Nikita Borisov, UC Berkeley

GSM phones use a number of algorithms, A3, A8, and A5 – A5 being the voice-encryption algorithm. All were designed in secret except for A5/0, although they are widely deployed (~100M units). A5/2 was reverse-engineered in August and a few hours later was broken. Borisov discussed how the algorithm works and how it was broken. The code to break the algorithm is written and functional, and it runs in record time. At the end of the talk, he made the source code for A5/1 and A5/2 available.

tracking hackers on IRC

by David Brumley

David Brumley works for the Stanford University Network Security Team (SUNSeT). He holds a Bachelor of Arts in mathematics with additional work in philosophy. In his free time he enjoys playing hockey and reading Kant.

<dbrumley@stanford.edu>

Few hackers are motivated purely by knowledge, science, and curiosity. Hackers continue to break into systems long after they become familiar with the technology. Instead, many continue to hack simply because of the social status it brings. For many, hacking *is* a social activity. Hackers meet online to discuss the latest hacking tools, their hacking conquests, and their personal lives. To be truly effective, system administrators and security professionals must become familiar with the social culture in which hackers dwell.

Internet Relay Chat (IRC) has replaced electronic bulletin boards as the social mecca for Internet addicts. Hackers are no exception. Cybersleuths must understand the jargon and tools used in this virtual society. By understanding IRC tools and jargon, a cybersleuth can determine the real identity of a hacker from birthplace to current address and telephone number.

System logs help administrators and security professionals track down criminals. They are useful as evidence that a crime has been committed, but not for much else. System logs show how and where the electronic bits came from, but they don't show *who* sent them. To prosecute successfully you must not only show where the intruder came from, but also who was physically using the keyboard at that particular time. IRC can be a tool for finding out.

For example, even with a full audit trail showing that an intruder came from a particular account on a particular ISP, the most you can hope to obtain is billing information for the account. While sometimes sufficient, this still hasn't shown you *who* was using the keyboard at the other end of the connection. The account you traced may have been stolen, set up with false billing information, or shared among several in a household.

An IRC-savvy administrator, however, may be able to determine the exact identity of the intruder. How? By listening to the hacker on IRC and reviewing configuration information on IRC tools left behind. A hacker's bragging about compromising your host is also a full confession, when logged. The IRC tools may be configured always to allow particular ISP connections, which may help in pinning down a hacker's location. With a little deductive reasoning you can pin down who hacked your machine, what his name is, where he lives, and even his favorite corner liquor store.

There are literally dozens of IRC networks. The most popular are DALnet, EFNet, and Undernet. Each IRC network is composed of hundreds, perhaps thousands, of channels where individuals with similar interests can chat realtime with one another. Channels are dynamic by nature. A channel is created the first time someone enters and destroyed when the last person leaves. The first person in a channel is also the channel operator, known as "chanops," or simply "ops." A channel operator is the superuser for the channel: she can invite other users to the channel, set the topic, decide who can talk, and give or take operator status from others on the channel.

On some networks, such as DALnet and Undernet, channels can be registered after creation. Registration allows the creator to become a channel operator every time he logs on to his channel. Registration assigns ownership of a channel.

Many IRC networks, including the ever popular EFNet, don't have channel registration. When you leave a channel, you leave all your privileges in that channel as well. You must be re-op'ed every time you join the channel. Hackers love dynamic networks like

these because it allows them to take over channels. Hackers will force all legitimate users out of a channel until they are all that is left. When they are the only ones in the channel, they can op themselves. The primary method for forcing users off a channel is a denial-of-service (DoS) attack. If the victim's computer is swamped by a DoS attack, it will time out and disconnect from IRC.

Hackers who participate in these dynamic IRC networks have one primary goal: to keep operator status on the channels they frequent. To do this they must protect against others trying to take over their channel and rogue administrators randomly de-op'ing them, and deal with the inevitable DoS attack. To solve these problems, hackers have come up with ingenious ways to create redundant connections from multiple hosts to an IRC network.

The simplest solution is to run multiple IRC clients, such as ircII or BitchX, from several hosts. By running the clients under screen(1), a UNIX terminal multiplexer, they can detach IRC sessions into the background and reattach to them later. Each session corresponds to one nickname on the IRC network. If one host goes down, the hacker can always reattach to a running session on another machine. Since each nickname has operator status, the whole scheme is redundant. It is up to the intruder, however, to maintain daily every IRC session on every host – a very labor-intensive activity.

IRC "bots," short for robots, solve the problem of "hands-on" administration. The purpose of a bot is to sit on IRC and monitor channels for events. In a very simple sense, bots are only automated IRC clients. Running standalone, a bot will automatically op friends as specified in the configuration file and provide some channel-misuse control. The true power of bots, however, is their ability to link together to form "botnets." Each bot on a botnet serves as a redundant backup, automatically op'ing friends and other bots, enforcing channel bans, and ensuring that a party line exists.

The purpose of a bot is to sit on IRC and monitor channels for events.

Each bot on the botnet is a node. The botnet administrator appoints a master node, with the rest becoming slaves. The master node is in charge of distributing botnet configuration information with each slave. After initial configuration, the botnet administrator need only change configuration information on the master. The master will then automatically take care of updating all the sub-nodes.

To add a bot to the network, only a simple and static configuration file that specifies the master is needed. Once the new bot starts up, it will automatically contact the master and pull over the requisite configuration information to become a node on the botnet. The advantage to a hacker is enormous. For each new account or system compromise, the hacker need only upload the actual executable and a simple configuration file. Once started, the new bot automatically downloads all information, including current lists of channels, friends, and users. The new bot will then also automatically update every time the configuration on the master is changed.

The most famous bot is eggdrop, available at <http://www.eggdrop.net>. It serves as a good model for the typical bot. Its configuration file is divided into three logical sections (sometimes in three separate files, sometimes merged into one): user information, channel information, and bot information.

Channel information can be recognized by the `channel add <channelname>` Tcl command. Following the channel name is a list of options to apply to that channel. A sample eggdrop channel file looks like:

Most often, though, a hacker's goal is to hide his real IP address in case someone is watching him.

```
channel add #myhacker {
  chanmode "+ismt"
  dont-idle-kick
  +userbans
  +protectops
}
```

The `chanmode` defines what mode the channel should be. In this particular example channel `#myhacker` is invite-only (`i`), secret (`s`), and moderated (`m`) so that only channel operators can talk; and only channel operators can change topics (`t`).

The last three items define eggdrop configuration variables. Entries that begin with a plus (`+`) will enable options; entries that begin with a minus (`-`) will disable options. Entries with neither a plus nor a minus simply define a variable, that is, make it true. In this particular example the bot will not kick idle users from `#myhacker`; it will let user operators (as opposed to other bots on the botnet) ban people and will automatically re-op de-oped users. For a full list of options, see the example configuration file that comes with the eggdrop distribution.

A user entry for an eggdrop bot generally consists of four lines. The first line always contains the nickname, password, and flags of the bot user. The remaining three lines all use the first two characters to identify the type of configuration information. Entries that start with a "`-`" list user identifiers. To a bot, a user's identity is not her nickname, but the username@hostname.domain.zone from which she is connecting.

A line that begins with "`:`" is a botnet configuration entry. It lists the `HOSTNAME:PORT` that the particular bot for that user will listen on. When two bots communicate, they use the port on the host listed.

Lines that begin with "`!!`" or "`.`" contain time-stamp information on the user. Entries beginning with "`!!`" are the channel name and time stamp the user was last seen by the bot. Entries with "`.`" are the modification time of the entry itself. All times are kept in UNIX epoch format.

User files often contain dozens of bot users. If you've found an eggdrop configuration file on a compromised host, chances are that most of the entries in the file are also compromised hosts or accounts. A quick note to the administrator of each domain explaining that you've found a hacker configuration file that references his domain is appropriate. You can also use the information in creating an MO (*modus operandi*) file for the hacker. People listed in the user file are often friends of the hacker (whom you may see in the future) or alternate nicknames the hacker may be using.

Here's an example of an eggdrop user file:

```
eleet lypmjwfp2ee fbs /0 0 0 0
- *!eleet@*.elaine.Stanford.EDU, *!eleet@*.myth.stanford.edu
: firebird.stanford.edu:60000
!! 895178133 #stanford
. {created 894412528}
```

Hackers will often not connect to IRC directly. By using a variety of hosts, a hacker can subvert a ban, trick others into thinking he is someone else, or connect to an IRC server that limits connections. Most often, though, his goal is to hide his real IP address in case someone is watching him.

A "bounce" program reads from one port and writes to another, that is, it is a proxy. The most famous bounce programs are BNC and WinGate. Both accept a TCP connection, connect to a destination, and then relay anything from the original connection to

the destination. The primary legitimate use for WinGate are SOCKS and TCP proxies to the Internet. Although WinGate can be configured to require a password, that is not standard practice. When a hacker has access to a WinGate, he can "bounce" through the WinGate server to hide his tracks.

BNC, the word "bounce" with the vowels removed, are UNIX-based proxies designed primarily for "bouncing" IRC traffic. Whereas a WinGate can proxy multiple ports, a BNC runs as a daemon listening to only one port. After accepting a connection, it too proxies information read on the original connection to a destination. In addition to simple proxying, the BNC configuration file allows for creating fake `ident` responses, configuring virtual hosts, and limiting the number of users who can use the bounce.

Since these processes run for extended periods of time, a hacker will often try to hide them from an administrator. If a hacker has superuser access and is skillful, she can hide any process from any administrator. Luckily, many hackers are sloppy or lazy. Often they will just change the name of the program to something innocuous. A local favorite seems to be *pine*. The hacker runs the process under the new name, hoping that the administrator will not notice.

Because hackers are adept at hiding process names, you should always be aware of the network connections your host generates. NetStat and `lsof` (<http://vic.cc.purdue.edu/pub/tools/lsof_4.45_W.tar.gz>) are good tools for monitoring local network connections. An administrator should also be wary of local processes, such as `./pine` or `./emacs`, binding to unusual ports. It's a safe bet that `pine` doesn't listen to port 6666 and write to `irc.erols.com`.

After you've confirmed that there's a hacker on your system who appears to be using IRC, consider setting up a network sniffer. (Please make sure you talk to your institution's legal department and are aware of all applicable laws.) Network dumps are valuable because little, if any, IRC activity is encrypted. Even if a hacker uses an encrypted client, such as SSH, to log in, the actual connection to the IRC server will most likely be in cleartext.

`tcpdump` (available from <ftp://ftp.ee.lbl.gov>) is the standard packet sniffer on most UNIX hosts. By default it only captures the first few bytes of every transaction – just enough to diagnose routing and network problems. When you're interested in logging entire sessions, it's important to read all available packet information. With `tcpdump`, the `-s` option controls how much data in each packet is collected. Consult your network MTU to determine the optimum setting. We use:

```
# /usr/sbin/tcpdump -n -s 1600 -F <FILTER FILE> -w tcpdump.<date>
```

A quick and easy way to view the dump is to use the UNIX command `strings(1)`. If too much information is picked up, you can separate your `tcpdump` file using:

```
# /usr/sbin/tcpdump -r tcpdump.<date> -w <output> <FILTER>
```

and then run `strings` again on the output file. For example, if you're only concerned with IRC traffic (which normally is on port 6667), use:

```
# /usr/sbin/tcpdump -r tcpdump.<date> -w irc.<date> dst port 6667
# strings irc.<date>
```

After gathering as much information about the hacker as possible through a packet dump and information from the various IRC configuration files, compile an MO file. The MO should contain information such as the hacker's preferred nickname and any variations used, any dial-ups used, any related incidents, and any personal information discovered.

Don't accept files from strangers, don't run untrusted IRC scripts, and never run commands you don't understand.

On several occasions I have picked up the exact age, name, and location of the hacker! This type of data is invaluable when contacting law enforcement and correlating various incidents. I've found plotting the information on a map is a good way to provide a quick reference of active hackers.

On a slow afternoon I have also been known to go back to the MO files and check to see who is on IRC. If I believe I see the same hacker I'll send a quick note to the administrator of the domain, alerting them to a potential problem. Sometimes it turns out to be nothing, but the message is always appreciated.

I use `ircii` (available from <http://www.irchelp.org/irchelp/ircii/>), the classic UNIX IRC client, and primarily connect to EFnet. (Macintosh and PC users should check out <http://www.irchelp.org> for a list of clients.) Generally the UNIX clients are safe as long as you use common sense. Don't accept files from strangers, don't run untrusted IRC scripts, and never run commands you don't understand. With most clients all IRC commands start with a forward slash (`/`). Everything else is a message sent to the channel.

After connecting, the first thing I do is start a log. With `ircII` (v. 4.4.), the command is:

```
/set log on
```

The logfile will be named `IrcLog`. To change names, type:

```
/set logfile <logfile name>
```

To look for a person, use the `who` command:

`/who -nick <nickname>` (looks for a particular nickname)

`/who -host <hostname>` (looks for anyone using a particular host)

Wildcards are allowed. However, users marked as "invisible" will only show if you specify their exact nicknames.

When checking IRC, be sure to look for all variations of the nickname. Hackers have the habit of logging in from a hacked site on a secondary nickname, while logged in with their primary nickname on their dial-in account. For example, perhaps there is a hacker who goes by the nickname "eleet". Querying IRC for `eleet` and `eleet_` might show:

```
* eleet        H* user@ppp-7.isp.net
* eleet_       H* root@www.companyname.com
```

Chances are that `www.companyname.com` has been hacked. Even more interesting is that the person who did it probably, though not certainly (as `ident` responses can be faked), is also using the dialup `ppp-7.isp.net`. One note of caution: when using IRC, use the `who` command, not `whois`.

```
/who -nick <nickname>
```

will give you information on the nickname.

```
/whois <nickname>
```

will give you more information but also notify the user that someone is querying his nickname.

Because of the number of hackers using IRC, it is often the target of criticism, but there are thousands of legitimate users who use the IRC networks daily. As in any other community, there will always be a criminal element. When hackers do use IRC, it allows the

administrator to monitor the criminal element and gain insight into their methodologies and habits. The acquisition of this knowledge can help system administrators, law enforcement, and security professionals track and prosecute hackers more effectively.

Resources

<http://www.eggxpress.com>
Information on eggdrop, BNC, and BitchX configuration files.

<http://www.eggdrop.net>
The home of eggdrop.

<http://www.irchelp.org>
Provides introductory documents and tutorials for using IRC.

<http://metalab.unc.edu/dbarberi/papers/chats>
Papers about the social perspective of IRC.

<http://www.efnet.net>
The popular EFNet, a dynamic IRC network and a favorite of many hackers.

<http://www.undernet.org>
The Undernet IRC network.

<http://www.newnet.net>
The NewNET IRC network.

<http://www.self-evident.org>
Dedicated to news on EFnet, including information on hackers and channels they frequent.

setting up a Linux firewall

by Juan Matus

Juan Matus gave up being a Yaqui sorcerer to work as a system administrator because the work is more challenging. He now manages a small mixed network of Linux and Microsoft systems.

donjuan@spirit.com

Open-source operating systems such as Linux and BSD provide the ability to filter IP traffic. What started as plain old packet filtering has advanced, and now some stateful packet-filtering features are appearing in some of the modules available with newer Linux kernels. This article describes the nuts and bolts of setting up a simple Linux-based firewall.

My target audience is people with small networks and a full-time (non-dialup) direct link to an ISP. (For a reasonable cost, you can have a directional antenna on the roof giving 2 megabits per second.) The network topology is very simple. The firewall is a PC with two network interface cards (NICs). One card connects to your local network, and the other connects to your ISP. All packets from the Internet have to pass through the firewall to get to your local network.

A firewall is only one component in your overall security plan. Gray[1] has a good general discussion of security, and there are also the classic firewall books (Cheswick[6] and Chapman[7]) for a deeper understanding of the issues.

Getting Started

When installing your firewall, load only the software you understand and disable any software that you don't understand. If you don't have time to understand all the details, then install open systems where the design has been reviewed and critiqued by many people worldwide. If you are forced to use Microsoft's black boxes, then beware. To sum it up, the KISS principle applies: Keep It Simple, Stupid!

Although OpenBSD or FreeBSD may be more secure, Linux works well. Linux has also got enough mindshare among technical managers that justifying its use has become routine.

You will want to use a newer version of Linux, such as Red Hat 6.0 on CD. This kernel has built-in support for `ipchains`. You may need to download and build `ipchains`, a package that includes the administrative commands with some other versions of Linux. (The HOWTO URL[3] has pointers for finding source.) For instructions on installing Linux, start browsing at <http://www.linux.org/>, or just boot from the CD and give it a whirl.

The firewall PC can be a surplus Pentium 200 or similar box with whatever network interface cards are available. Linux will work with many network card types, but not all. You will have best luck with newer 3COM cards or those based on the DEC Tulip chipset. PCI cards are preferred because you don't have to assign IRQs (generally). Your hard disk should be large enough for the log files, at least 1 or 2 GB.

My examples assume that your local network has the address 192.168.0.0, because this is reserved for networks that are not connected to the Internet. If a packet from your local network gets out onto the Internet, routers will drop any responses back to your network. Also, no one can route a packet to you from the Internet (although someone located at your ISP can certainly attempt to route a packet to your internal network). As you install Linux, let's assume you put in 192.168.0.15 for the IP address and 255.255.255.0 for the netmask for the internal interface, and the address and netmask assigned by your ISP for the external interface. Viewed from the back (or top in the case of a tower) the first network PCI card will be on the left (top).

When Linux has been installed, the kernel should detect the network cards at boot time. The kernel will write some details to the screen. This information also goes to `/var/log/messages`. Use `ifconfig -a` to check that the cards are up, and check the activity lights on the cards. The interfaces will be called `eth0` and `eth1`. I will assume that you have `eth0` on your LAN and `eth1` connected to the ISP.

OS Configuration

Log in as root with the X11 window system (if you installed it). Shut down all unneeded services by editing `/etc/inetd.conf` and commenting out most lines in the file. You might consider commenting everything out, then disabling `inetd` itself. Go into the `/etc/rc.d/` directory tree and disable any unneeded processes and anything that you don't understand. You can disable processes using the GUI-based "Runlevel Editor," or you can go into `rc5.d` and change the links by hand.

Then reboot and try things out. If you have disabled too much (for example, `S10network`), X11 won't work anymore, but you can still use the alternate consoles (Alt and function keys) to log in. You will need networking, but not `S15netfs` or `S11portmap`.

You can use the `ps` command to list running processes, or use `netstat` to list listening Internet services:

```
[root@firewall]# netstat -aep –inet
...
tcp   0  0 localhost:domain    *:*   LISTEN   root    732  537/named
tcp   0  0 *:ftp               *:*   LISTEN   root    716  523/inetd
...
```

There will be some processes associated with the `LISTEN` state on a port. Are there any that you did not expect? The DNS process is named, and it is probably okay. However, you probably want to disable the ftp server. The `sunrpc portmap` process is mostly needed by NFS, and you disabled that, so let's make sure you disable portmap too.

You might want to disable `sendmail` until you understand it better. `sendmail` is much more secure now than it was in the past, and maybe it could be left enabled. If you have the time, you can install an alternate mail package such as `exim` (<http://www.exim.org/>), `qmail` (<http://www.qmail.org/>), or `postfix` (<http://www.porcupine.org/postfix-mirror/start.html>). For our example, we have disabled `sendmail` on the firewall because we will use port forwarding to direct SMTP connections to an internal server.

There is a new window system called GNOME running on top of X11. GNOME components open sockets, and in the spirit of disabling anything you don't understand, you will want to shut down GNOME. The window-manager menus will let you select the alternative FVWM. (I can imagine the GNOME developers getting steamed up at this point, as they have put much work into a really impressive desktop. Sorry, we are setting up a firewall, not a workstation.)

In the menus you will find a simple GUI tool called "Network Configurator" to help assign IP addresses and netmasks. It just changes the text file `/etc/sysconfig/network`, and any changes take effect on a reboot. You need to enable IPV4 forwarding and set the IP address for `eth1` (if you didn't do this during installation). Let's assume that your ISP has assigned you the 223.56.222.4 network with a netmask of 255.255.255.252. Let's assume that your `eth1` address must be set to 223.56.222.6. Your ISP has set the Domain Name System to point `yourcorp.com` to the latter IP address. With such a restrictive netmask, that box that connects to the rooftop antenna must respond to 223.56.222.5. The latter number must be put in the Gateway field of the `Sysconfig` tool. If you have a direct connection, you will use the address of the router at your ISP's end of the connection.

When installing your firewall, load only the software you understand and disable any software that you don't understand.

The socks daemon will make the appropriate connection across the Internet.

Outgoing Connections

There are two approaches to firewalling. You can filter at the packet level or at the application level. In the latter case, you set up a proxy server on the firewall, and all connections are made via this server. The proxy server can take better control of the connections, and you can set up user-authentication mechanisms if desired. However, the server can be more trouble to set up than a simple packet filter, and performance will suffer slightly. For connection speeds less than 10MBs and small networks, performance should never be a problem. Both approaches, packet filtering and proxy server, will be discussed below.

For outgoing HTTP and FTP connections, we can run a socks proxy server on the firewall. All the hosts on the local network will connect to port 1080 (`socks`) on the firewall, and the `socks` daemon will make the appropriate connection across the Internet. The (hidden) local hosts will appear on the Internet as if they were at 223.56.222.6.

You can get `socks5-v1.0r9.tar.gz` from <http://www.socks.nec.com/cgi-bin/download.pl> and build it. You will need a startup file in `/etc/rc.d/init.d/socks`:

```
#!/bin/bash
# config: /etc/socks5.conf
# source function library
. /etc/rc.d/init.d/functions
case "$1" in
start)
export SOCKS5_V4SUPPORT=1
echo -n "Starting socks5: "
daemon /bin/socks5 -p
echo
;;
stop)
echo -n "Shutting down socks5: "
kill `cat /tmp/socks5.pid-1080`
echo
;;
restart)
$0 stop
$0 start
;;
*)
echo "Usage: socks {start|stop|restart}"
exit 1
esac

exit 0
```

The `SOCKS5_V4SUPPORT=1` flag is needed because the currently available Netscape and Internet Explorer browsers support only version 4 of the protocol. Go to `rc3.d` and link `S25socks` to `/etc/rc.d/init.d/socks`. Do the same in `rc4.d` and `rc5.d`.

Now edit the `/etc/socks5.conf` file:

```
# A socks5 Config file for a dual homed server
#
# interface dest-host dest-port interface-address
interface 192.168.0. - eth0
interface - - eth1
#
# auth source-host source-port auth-methods
auth - - n,u
#
# permit auth cmd src-host dest-host src-port dest-port [user-list]
permit - - 192.168.0. - - -
```

The dashes are wildcards, so they will accept any value.

Now go to each host on the local network and set the browser options so it uses the `socks` proxy server (port 1080 on host 192.168.0.15). Also set each host to use DNS on your firewall. Now you should be able to use your browser to access HTTP and FTP sites. Note that a standalone FTP client program will not work unless it knows about the `socks` proxy. There is a kernel module that can support FTP clients through the use of `ipchains` (see Russell's IPCHAINS-HOWTO[3]). You can also get also socksified `telnet`, `finger`, `ping`, `traceroute`, and `archie`.

In the Linux world, it is known as Masquerading with Port Forwarding.

Incoming Connections

You will want to permit machines on the Internet to connect to some services on hosts on the local network. For example, you might permit SMTP connections to a mail host on the local network. However, you want to block connections to most services. When connections are permitted, they will be to the firewall's externally visible IP address (223.56.222.6), and the firewall will rewrite each incoming packet with the address of a host on the local network. Packets flowing in the return direction will be rewritten to appear to come from the firewall.

In the RFCs, this is known as Network Address Translation (NAT). In the Linux world, it is known as Masquerading with Port Forwarding. Performance will be excellent because the packet processing is done by networking code in the kernel.

The Red Hat 6.0 kernel is already configured correctly for our purposes, with the exception of the following patch. You will need to build and install `ipmasqadm-0.4.2.tar.gz` from < http://juanjox.kernelnotes.org/>. This includes some tools used in port forwarding, which is explained below.

The `ipchains` command has three chains, one named `input` for packets destined to the firewall, one named `output` for packets originating at the firewall, and one named `forward` for packets being transferred across the firewall (as if the system were a router, forwarding packets between networks). Let's add some lines to `/etc/rc.d/rc.local`. The syntax of the `ipchains` command is complex, so I hope the following working example helps you.

```
# clear the forward chain
/sbin/ipchains -F input
/sbin/ipchains -F forward
/sbin/ipchains -P forward DENY
```

The `-F` flags flush existing rules from the named chains. The `DENY` rule will drop packets silently. For debugging, you might want to temporarily replace it with a `REJECT` rule, which will return a `RESET` instead of silently dropping packets.

```
#for debugging:
#/sbin/ipchains -P forward REJECT
```

The `-P` flag establishes a policy, that is, `REJECT` any forwarded packets by default. Now, let's add lines to set up forwarding to an internal email server in `/etc/rc.d/rc.local`:

```
# forward incoming packets to the mail server with masq
server=mailhost
servip=192.168.0.27/32
/sbin/ipchains -A forward -p tcp -s $servip 25 -j MASQ
```

The `-A` flag appends this rule to the forward chain. The arguments set up the protocol (`-p tcp`), server name/address (`-s $server`), and port address (25 for `sendmail`), and enable masquerading (`-j MASQ`). This rule is needed for the port forwarding below.

When you have the rules right, any log entries could show rogue packets.

```
# allow outgoing e-mail
/sbin/ipchains -A forward -p tcp -s 192.168.0.27 -d 0/0 25 -j MASQ
```

This rule allows outgoing connections from the mail server at 192.168.0.27 to port 25 of any host on the Internet. Connections will appear to come from the firewall.

```
# log anything which does not match the above rules
/sbin/ipchains -A forward -l
```

This rule causes `ipchains` to write to the log when an unwanted packet is about to be dropped. This can help you debug your rules. When you have the rules right, any log entries could show rogue packets. This sort of logging is a mixed blessing. If you don't check `/var/log/messages` regularly, an attempted break-in can easily be overlooked.

You can add more `ipchains` rules as desired, perhaps rules to log any unwelcome port scanning. Now, the port-forwarding rules:

```
# flush any existing rules
ipmasqadm portfw -f
# forward incoming mail connections to our server
ipmasqadm portfw -a -P tcp -L internet 25 -R $server 25
# list the rules
ipmasqadm portfw -l
```

The `-L internet` must match a line in the `/etc/hosts` file defining `internet` as the IP address of the `eth1` interface, 223.56.222.6.

The `portfw` rules work with a previous `MASQ` rule to permit incoming connections to the mail host at 192.168.0.27. Normally, the `MASQ` rule causes an outgoing connection to be rewritten for masquerading. The `portfw` rule turns the `MASQ` rule around so that the connection can be incoming.

You should test this from a host somewhere on the Internet. Do something like

```
telnet yourcorp.com 25
```

You should now see SMTP responses from your mail host. Incoming mail should now get through, assuming that your mailhost is configured correctly.

Resources

[1] First, read Bob Gray's article on security. <http://boulderlabs.com/7.security>

[2] Firewall and Proxy Server HOWTO by Mark Grennan. <http://www.linuxdoc.org/HOWTO/Firewall-HOWTO.html>

[3] Linux IPCHAINS-HOWTO by Paul Russell. <http://www.rustcorp.com/linux/ipchains/HOWTO.html>

[4] The Linux IP Masquerade Resource. <http://members.home.net/ipmasq/>

[5] The X/OS Linux firewall site. <http://www.xos.nl/linux/ipfwadm/paper/linuxfw2.html>

[6] Bill Cheswick and Steven Bellovin, *Firewalls and Internet Security*. Reading, MA: Addison-Wesley, 1994. (Contains an extensive bibliography.)

[7] Brent Chapman and Elizabeth Zwicky, *Building Internet Firewalls*. Sebastopol: O'Reilly & Associates, 1996.

[8] `socks5` configuration samples. <http://www.socks.nec.com/s5examples.html>

There are many Linux Web sites for firewalling and security. You can spend many hours following links from the sites mentioned above.

SENSS Bruce

Developing a Tool to Aid Intranet Security

In recent years, the network-vulnerability scanner (NVS) has become a favored tool in the network-security manager's arsenal; programs like ISS, NMap, and Sun's (internal) AutoHack provide valuable feedback for securing Intranet hosts.

by Alec Muffett
Alec Muffett is a senior staff engineer for Sun Professional Services EMEA.
<Alec.Muffett@UK.Sun.COM>

These tools typically suffer from one problem – the inability to provide a comprehensive security report for a host *as seen from the inside.* An NVS can tell you whether a particular revision of a buggy network daemon is installed on a given host, but it is less likely to be able to tell you that the host's `/etc` directory has `rwxrwxrwx` permissions. This contrasts with the effectiveness of older host-vulnerability scanner (HVS) tools such as COPS, which – because they run *inside* a host – can provide a thorough report of almost all the security weaknesses a machine exhibits.

For several years, people have tried to marry these two kinds of functionality, with varying degrees of success. The obstacles have been:

- NVS systems can have problems scaling to WAN-sized networks and typically only give a view of the security of a network of hosts as seen from *one place* in the network – the auditing host.
- HVS systems give a much better analysis of host weaknesses but typically are installed only *once* on any given host, going rapidly out of date, so that newly discovered bugs cannot be tested for without upgrading the HVS on every host in the network; also, there are major problems related to securely collecting and indexing reports from many HVS systems.

In 1997, a small SunLabs team decided to attempt to address these problems. The result has become the primary component of the Sun Enterprise™ Network Security Service, called Bruce – a networked host-vulnerability scanner.

How Bruce Works

After long debate, we decided to implement a network of service daemons to be installed on every host on an intranet, interlinked in a *web of trust*, formalizing the *trusted host* model of computer security that is familiar to most system administrators.

We chose to implement the daemons and driver code in Java, since this provided the opportunity for a platform-independent code base, as well as strong typing, RMI-based remote execution, rapid coding with reduced opportunity for buffer overflows, and access to activation mechanisms and cryptography hooks.

The only problems that we encountered in using Java to implement Bruce were:

- Waiting for cryptographic (Java2) functionality on many platforms. Lacking the immediate ability to run the code everywhere was irksome, but we viewed this as less a problem than a waiting game, and we seem to be being proven correct, now that Java2 is widespread.
- Platform-independent languages are weak for diagnosing and fixing platform-specific bugs. It should have struck us as obvious at the outset that a language with no notion of, for example, `setuid()` would have problems manipulating permissions on UNIX.

With the latter issue, after experimenting with JNI native-method extensions, we chose to maintain a platform-independent code base and adopt the following philosophy: *Use Java for secure, platform-independent glue; use platform-specific tools for platform-specific tasks.*

From these, we developed a design for a six-stage system that would do what we wanted:

1. Run a daemon on each host.

2. Command that daemon to launch some named task that has some security-checking function, for example `OSPatchRevisionCheck`; we call these tasks *pollets* – originally *policy applets* – in order to emphasize the platform-independent nature of the task names.

3. Have the daemon consult a table – embedded as part of the pollet launch request – that maps the `OSPatchRevisionCheck` pollet into the name of a package that implements that pollet for the host, selecting the package on the basis of the host's OS name, OS revision, CPU type, and fully qualified domain name.

4. The daemon then unpacks the package and executes its content, generating report output that is stored locally.

Pollet Name	OS	OS Rev	Arch	Hostname	Type	Package Name	Options
NFSSrvChk	sunos	5.*	sparc	*	package	sol-nfssrv-cfg	
OSPatchRevChk	sunos	5.*	sparc	*	package	sol-sparc-pchk	
OSPatchRevChk	sunos	5.*	sparc	*.uk.sun.com	package	uk-sol-pchk	tmp=/var/tmp
OSPatchRevChk	sunos	5.5.1	sparc	coyote.uk.sun.com	package	uk.coyote-pchk	tmp=/var/tmp
OSPatchRevChk	linux	2.*	i?86	{foo,bar}.uk.sun.com	package	linux-x86-pchk	

An example table, with wildcards, mapping between pollets and pollet-packages.

5. The person who launched the command can reconnect later to retrieve the output.

This is a simplified view of how Bruce operates; so far, it makes Bruce sound like little more than an advanced HVS and doesn't give much of a clue how to make this into a *networked* tool. We then add a further stage:

6. One of the pollets, `Dispatch`, consults a (possibly empty) list of *child* hosts, held on each machine, and uses it to further punt the launch request onward to each of those children, polling them later to retrieve the output they generate. The `Dispatch` pollet is also responsible for propagating new revisions of pollet packages to the children, providing a shrink-wrapped software-distribution mechanism.

Thus, by simple recursion, a pollet launch request is propagated among a tree of hosts, rooted at some suitably trusted parent. Output from child hosts is stored in the same manner as output generated locally, so that all output from all hosts eventually percolates up to the tree root. Because parent/child relationships are maintained locally (i.e., grandparents don't know in advance who their grandchildren will be), it is simple to add hosts into a Bruce Network without centralized intervention and associated administrative loading.

A pollet's output consists of a serialized Java object containing chunks of HTML and other WWW content, held in a miniature filesystem; it is expected that this will gradually migrate to XML to allow mechanized processing of pollet output. A simple HTTP server embedded in the Bruce daemon provides the hook for report-browsing by any Web browser, including `lynx`.

Obviously, a simple Web of trust is not enough to ensure security of a Bruce Network; thus a X.509 *master certificate* is installed on each host, the private key of which is used

to authenticate a chain of other certificates and keys that eventually digitally sign and authenticate launch requests, preventing forgery. Other digital signatures protect the pollet-mapping tables and packages against tampering, and each launch request contains a 64-bit random serial number that is used to prevent replay attacks and loops in the Bruce Network.

Further, Bruce provides hooks to extend Java's remote method invocation (RMI) mechanism, using *pluggable secure transport* (PST) modules to provide strong authentication, encryption, or compression between pairs of hosts; the selfsame configuration files for these PSTs also automatically generate TCP access-control lists, limiting undesired network access to the sockets used by Bruce.

Bruce cannot provide trustworthy results for a host that has already been compromised.

What We Didn't Try to Solve and Why Not

A great deal of pragmatism was brought to bear on the design of Bruce. Certain problems were deemed insoluble or political. Others were addressed simply:

- To reduce denial-of-service and scalability problems, all communication is initiated from *parent* to *child*. This means that all output retrieval must be made via *polling*, which is slow but immensely simplifies the security design.
- Due to US export restrictions, the Web server interface is currently implemented as an HTTP daemon bound to 127.0.0.1, rather than HTTPS. With recent export-control relaxations, the option of upgrading to HTTPS will be investigated. Similarly, the core PST modules currently use freely exportable digital-signature algorithms and do not provide cryptosecrecy.

Other problems were more subtle. Suggestions of encrypting or signing pollet outputs were considered, but eventually rejected; without use of export-controlled public-key encryption, it would be impossible to sign or encrypt the pollet output without use of a private key that would have to be stored on each host, or embedded *en clair* in the launch request. Simple obfuscation of this key would add nothing to security, being merely security through obscurity.

Following this premise, we also rejected attempts to solve man-in-the-middle attacks, where a Trojan-horse version of Bruce is installed on some intermediate host, forging pollet output for itself and child hosts. If a system has been compromised to this extent, then no trustworthy mechanism is available to check, for example, Bruce-executable code signatures.

Instead, we cast a different light upon the software:

- Bruce is a tool that is meant to aid the upkeep and maintenance of *existing* integrity in an intranet.
- Like all security software, Bruce cannot provide trustworthy results for a host that has already been compromised; however, this does not render it useless.

The benefits of Bruce come from its ability to scale; in a network of many hosts, by using Bruce in a sensibly configured manner – a shallow tree, with hardened intermediate hosts – an administrator is equipped to check and fix every machine with a single operation, whereas a hacker must still compromise them one at a time.

Summary

SENSS Bruce is a tool that provides an intranet administrator with the ability to maintain the security of each host on a network, also permitting security checking in a platform-independent manner. Its aim is to provide this functionality *without making the security of a machine any worse than it already is.* Its benefit is that it allows system

SENSS Bruce will be made available (including source code) to the Internet community at large.

administrators to bulk-administer entire networks in a secure, batch-like manner, fixing them before the hackers can get to them.

Availability

SENSS Bruce will be made available (including source code) to the Internet community at large under the terms of the Sun Community Source License. More information pertaining to Bruce may be requested by sending email to <bruce-feedback@sun.com>.

Resources

AutoHack

<ftp://coast.cs.purdue.edu/pub/doc/network/muffett-wanhack.ps.Z>
<ftp://coast.cs.purdue.edu/pub/doc/network/muffett-wanhack-slides.tar.gz>

COPS

<http://www.cs.purdue.edu/homes/spaf/tech-reps/993.ps>

ISS

<http://www.iss./net/>

Java

<http://www.sun.com/java/>

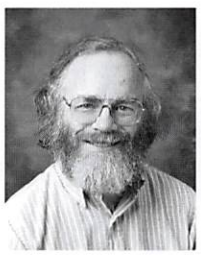
NMAP

<http://www.insecure.org/nmap/>

Secure Java Coding Web pages

<http://www.javaworld.com/javaworld/jw-12-1998/jw-12-securityrules.html>
<http://www.securingjava.com>

distributed firewalls



by Steven M. Bellovin

Steven Bellovin, who as a graduate student helped create netnews, is a Fellow at AT&T Labs Research. He is the co-author of *Firewalls and Internet Security: Repelling the Wily Hacker.*"

<smb@research.att.com>

Conventional firewalls[5] rely on the notions of restricted topology and controlled entry points to function. More precisely, they rely on the assumption that everyone on one side of the entry point – the firewall – is to be trusted, and that anyone on the other side is, at least potentially, an enemy. The vastly expanded Internet connectivity in recent years has called that assumption into question. So-called "extranets" can allow outsiders to reach the "inside" of the firewall; on the other hand, telecommuters' machines that use the Internet for connectivity need protection when encrypted tunnels are not in place.

Other trends are also threatening firewalls. For example, some machines need more access to the outside than do others. Conventional firewalls can do this, but only with difficulty, especially as internal IP addresses change. End-to-end encryption is another threat, since the firewall generally does not have the necessary keys to peek through the encryption.

Some people have suggested that the proper approach is to discard the concept of firewalls. Firewalls, they feel, are obsolete, or are not needed if cryptography is used. We disagree.

Firewalls are still a powerful protective mechanism. Most security problems are due to buggy code – in 1998, 9 of 13 CERT advisories concerned buffer overflows, and 2 others were cryptographic bugs – and cannot be prevented by encryption or authentication. A firewall shields most such applications from hostile connections.

Firewalls are also useful for protecting legacy applications. Applications that require strong authentication should in principle provide their own, but many older protocols and implementations do not. Saying that strong cryptography should be used is true but irrelevant; in the context of such applications, it is simply unavailable.

More subtly, firewalls are a mechanism for *policy control.* That is, they permit a site's administrator to set a policy concerning external access. Just as file permissions enforce an internal security policy, a firewall can enforce an external security policy.

To solve these problems while still retaining the advantages of firewalls, we propose a distributed solution. In such a scheme, policy is still centrally defined; enforcement, however, takes place on each endpoint. We thus retain the advantages of firewalls while avoiding most of the problems we have described, most notably the dependency on topology.

The concept of distributed firewalls rests on three notions:

- a *policy language* that states what sort of connections are permitted or prohibited
- any of a number of *system-management tools*, such as Microsoft's SMS or ASD[7]
- IPSec[6], the network-level encryption mechanism for TCP/IP.

The basic idea is simple. A compiler translates the policy language into some internal format. The system-management software distributes this policy file to all hosts protected by the firewall. And incoming packets are accepted or rejected by each "inside" host, according to both the policy and the cryptographically verified identity of each sender.

Angelos Keromytis, Matt Blaze, and John Ioannidis made a number of useful suggestions, especially with regard to KeyNote.

Policies and Identifiers

Many possible policy languages can be used, including file-oriented schemes similar to Firmato[4], the GUIs that are found on most modern commercial firewalls, and general policy languages such as KeyNote[1, 14]. The exact nature is not crucial, though clearly the language must be powerful enough to express the desired policy. A sample is shown in Figure 1.

```
inside_net = x509{name="*.example.com"};
mail_gw = x509{name="mailgw.example.com"};
time_server = IPv4{10.1.2.3};

allow smtp(*, mail_gw);
allow smtp(mail_gw, inside_net);
allow ntp(time_server,inside_net);
```

Figure 1. A sample policy configuration file. SMTP from the outside can reach the machine only with a certificate identifying it as the mail gateway; it, in turn, can speak SMTP to all inside machines. NTP – a low-risk protocol that has its own application-level protection — can be distributed from a given IP address to all inside machines. Finally, all outgoing calls are permitted.

What is important is how the inside hosts are identified. Today's firewalls rely on topology; thus, network interfaces are designated "inside," "outside," "DMZ," etc. We must abandon this notion, since distributed firewalls are independent of topology.

A second common host designator is IP address. That is, a specified IP address may be fully trusted, able to receive incoming mail from the Internet, and so on. Distributed firewalls can use IP addresses for host identification, though with a reduced level of security.

Our preferred identifier is the name in the cryptographic certificate used with IPSec. Certificates can be very reliable unique identifiers. They are independent of topology; furthermore, ownership of a certificate is not easily spoofed. If a machine is granted certain privileges on the basis of its certificate, those privileges can apply regardless of the machine's physical location.

In a different sense, policies can be "pulled" dynamically by the end system. For example, a license server or a security-clearance server can be asked if a certain communication should be permitted. A conventional firewall could do the same, but it lacks important knowledge about the context of the request. End systems may know such things as which files are involved and what their security levels might be. Such information could be carried over a network protocol, but only by adding complexity.

Distributed Firewalls

In a typical organizational environment, individuals are not necessarily the administrators of the computers they use. Instead, to simplify system administration and to permit some level of central control, a system-management package is used to administer individual machines. Patches can be installed, new software distributed, and so forth. We use the same mechanisms, which are likely to be present in any event, to control a distributed firewall.

Policy is enforced by each individual host that participates in a distributed firewall. The security administrator – who is no longer necessarily the "local" administrator, since we are no longer constrained by topology – defines the security policy in terms of host identifiers. The resulting policy (probably, though not necessarily, compiled to some convenient internal format) is then shipped out, much like any other change. This policy file is consulted before processing incoming or outgoing messages, to verify their compliance. It is most natural to think of this happening at the network or the transport layer, but policies and enforcement can equally well apply to the application layer. For example, some sites might wish to force local Web browsers to disable Java or JavaScript.

Policy enforcement is especially useful if the peer host is identified by a certificate. If so, the local host has a much stronger assurance of its identity than in a traditional firewall situation. In the latter case, all hosts on the inside are in some sense equal. If any such machines are subverted, they can launch attacks on hosts they would not normally talk

to, possibly by impersonating trusted hosts for protocols such as `rlogin`. With a distributed firewall, though, such spoofing is not possible; each host's identity is cryptographically assured.

This is most easily understood by contrasting it to traditional packet filters[9]. Consider the problem of electronic mail. Because of a long-standing history of security problems in mailers, most sites with firewalls let only a few designated hosts receive mail from the outside. They in turn will relay the mail to internal mail servers. Traditional firewalls would express this by a rule that permitted SMTP (port 25) connections to the internal mail gateways; access to other internal hosts would be blocked. On the inside of the firewall, though, access to port 25 is unrestricted.

With a distributed firewall, all machines have some rule concerning port 25. The mail gateway permits anyone to connect to that port; other internal machines, however, permit contact only from the mail gateway, as identified by its certificate. Note how much stronger this protection is: even a subverted internal host cannot exploit possible mailer bugs on the protected machines.

Distributed firewalls have other advantages as well. The most obvious is that there is no longer a single chokepoint. From both a performance and an availability standpoint, this is a major benefit. Throughput is no longer limited by the speed of the firewall; similarly, there is no longer a single point of failure that can isolate an entire network. Some sites attempt to solve these problems by using multiple firewalls; in many cases, though, that redundancy is purchased only at the expense of an elaborate (and possibly insecure) firewall-to-firewall protocol.

A second advantage is more subtle. Today's firewalls don't have certain knowledge of what a host intends. Instead, they have to rely on externally visible features of assorted protocols. Thus, an incoming TCP packet is sometimes presumed legitimate if it has the "ACK" bit set, since such a packet can only be legitimate if it is part of an ongoing conversation – a conversation whose initiation was presumably allowed by the firewall. But spoofed ACK packets can be used as part of "stealth scanning." Similarly, it is hard for firewalls to treat UDP packets properly, because they cannot tell if they are replies to outbound queries and hence legal, or if they are incoming attacks. The sending host, however, knows. Relying on the host to make the appropriate decision is therefore more secure.

This advantage is even clearer when it comes to protocols such as FTP[11]. By default, FTP clients use the `PORT` command to specify the port number used for the data channel; this port is for an incoming call that should be permitted, an operation that is generally not permitted through a firewall. Today's firewalls – even the stateful packet filters – generally use an application-level gateway to handle such commands. With a distributed firewall, the host itself knows when it is listening for a particular data connection and can reject random probes.

The most important advantage, though, is that distributed firewalls can protect hosts that are not within a topological boundary. Consider a telecommuter who uses the Internet both generically and to tunnel in to a corporate net. How should this machine be protected? A conventional approach can protect the machine while it's tunneled. But that requires that generic Internet use be tunneled into the corporate network and then back out to the Internet. Apart from efficiency considerations, such use is often in violation of corporate guidelines. Furthermore, there is no protection whatsoever when the tunnel is not set up. By contrast, a distributed firewall protects the machine all of the time, regardless of whether or not a tunnel is set up. Corporate packets, authenti-

The most important advantage is that distributed firewalls can protect hosts that are not within a topological boundary.

In a hybrid implementation, some hosts are behind a traditional firewall, while other hosts live on the outside.

cated by IPSec, are granted more privileges; packets from random Internet hosts can be rejected. And no triangle routing is needed.

Hybrid Firewalls

Remote Nodes and IPSec

Although a full implementation of distributed firewalls is the most secure and the most flexible, hybrid implementations can exist. That is, one can combine the techniques described here with traditional firewalls, achieving adequate functionality at lower cost, especially until IPSec support becomes ubiquitous.

In a hybrid implementation, some hosts are behind a traditional firewall, while other hosts live on the outside. An IPSec gateway at the central site provides connectivity to the outside machines. (Whether this gateway is inside the traditional firewall, outside it, in parallel with it, or even integrated with it is largely irrelevant to this discussion.) This configuration is common at companies with a major central site and some number of telecommuters.

As in ordinary virtual private networks (VPNs), remote hosts have full access to the inside, by virtue of the IPSec tunnel. Traffic from inside machines to the remote nodes is similarly protected. What is distinct is that traffic from remote nodes to the rest of the Internet is governed by the central site's security policy. That is, the firewall administrator distributes a security policy to the remote nodes, as we have described. Ideally, of course, this same policy statement is used to control the traditional firewall, thus ensuring a consistent security policy.

Distributed Firewalls and Topological Knowledge

Another variety of hybrid implementation ignores IPSec entirely. In this situation, address-dependent policy rules are distributed to, and enforced by, every individual host within a site. (Many newer systems support such functionality in the kernel.) While address-based authentication is quite weak, if a simple router prevents address-spoofing from the outside, the security should be comparable to that of traditional firewalls.

Here, we use system-management techniques to ensure consistent policy. We also rely on topology, thus forfeiting the ability to protect remote hosts. However, we still eliminate the single chokepoint and point of failure.

A final hybrid scheme combines the two previous hybrid schemes. Again, a simple router prevents address-spoofing by outside machines that talk to inside nodes. IPSec is used to tunnel traffic from inside nodes to remote nodes. On these systems, the IPSec module provides anti-spoofing protection. Finally, all protected machines, whether local or remote, receive and enforce an address-based firewall policy.

Application-specific Tunnels

With the proper sort of routing, triangle routing can be used for a few protocols, while distributed-firewall techniques are used for most. Suppose, for example, that some protocol requires an application-level proxy for proper firewalling. Packets for that protocol can be routed through an IPSec-protected tunnel to the inside of the firewall. From there, they can be forwarded to the Internet, but only after proper processing by the firewall.

In the extreme case, of course, this reduces to a conventional firewall. But if few enough protocols need this sort of filtering, it becomes an attractive variant.

Implementation Techniques

A number of different techniques can be used to implement distributed firewalls. The strongest is to use end-to-end IPSec. Each incoming packet can be associated with a certificate; the access granted to that packet is determined by the rights granted to that certificate. Consider a packet destined for port 25 (SMTP) on an inside machine, given the sample policy shown in Figure 1. If the certificate identifies the source as `mailgw.example.com` – note that this is not necessarily the same as the machine with that domain name – the packet will be accepted. If the certificate name is different (for clarity, we omit any discussion of the certificate's signature chain, though that is clearly important for real use) or if there is no IPSec protection, the packet will be dropped as unauthorized.

We note that the necessary filtering is prescribed by the IPSec architecture[6]. Specifically, the inbound Security Policy Database (SPD) is used to reject illegal input packets, while the outbound SPD can be used to control outgoing connections. An informal survey showed that most commercial IPSec implementations either support port number–granularity security associations or will in the near future.

Application-level protection can be achieved by distributing application-specific policy files. Thus, Web browsers can be told by the central site to reject, for example, all ActiveX controls or Java applets. Note that this is a hard problem for conventional firewalls[10]; doing it on the end hosts is more secure, if the policy-distribution problem can be solved.

The hardest problem is handling protocols such as FTP without touching the application. That is done most easily with per-process keying for IPSec. For example, a policy rule for FTP would indicate that outbound connections to port 21 must be protected with IPSec, and that all other TCP connections protected by that security association are legal. Since only that process can use that security association, and it would only have the FTP data channel open, an adequate level of protection can be achieved.

KeyNote is an especially attractive choice for a policy language. Indeed, Blaze, Ioannidis, and Keromytis[14] explicitly note its suitability for configuring packet filters. Its advantages include the integration of credentials with policy specification, and an ability to use a single mechanism to specify policy at different levels.

In some sense, distributed firewalls are similar to host-based enforcement as implemented in TCP wrappers[12], a similar package for restricting access to assorted daemons[8], or various commercial PC "personal firewall" packages. However, those schemes are still dependent on IP addresses (and hence topology) and do not include central policy definition as an integral piece.

Change Management

Given that access rights in a strong distributed firewall are tied to certificates, the access rights can be limited by changing the set of certificates accepted. Suppose, for example, that a security flaw is discovered in some networked application. A new certificate can be distributed to hosts in the same distribution as the patch. Only hosts with newer certificates are then considered to be "inside"; if the change is not installed, the machine will have fewer privileges (Figure 2).

```
inside_net = x509{name="*.example.com", ver>19990315};
mail_gw = x509{name="mailgw.example.com"};
time_server = IPv4{10.1.2.3};

allow smtp(*, mail_gw);
allow smtp(mail_gw, inside_net);ntp(time_server,inside_net);
allow *(inside_net, *);
```

Figure 2. A policy configuration file with version information.

In some environments, it may even be possible to use certificates to help ensure user compliance. A PolicyMaker certificate[2,3] could contain code that checked the configuration of the user's system. If the check failed – that is, if the proper versions of the proper files were not present – the certificate would not allow itself to be used for authentication. (Clearly, without tamper-resistant hardware this is not an absolute check, but it can help protect against accidental misconfiguration and less-determined malfeasance.)

The certificate version mechanism also protects against new, insecure machines that are installed on an inside network. Until the appropriate filtering software and rulesets are installed (and possibly until the machine is otherwise made secure by the organization's administrators), no certificate is issued. The machine is thus an outside machine, regardless of its physical location.

Threat Comparison

Distributed firewalls have both strengths and weaknesses when compared to conventional firewalls. By far the biggest difference, of course, is their reliance on topology. If your topology does not permit reliance on traditional firewall techniques, there is little choice. A more interesting question is how the two types compare in a closed, single-entry network. That is, if either will work, is there a reason to choose one over the other?

Service Exposure and Port Scanning

Both types of firewalls are excellent at rejecting connection requests for inappropriate services. Conventional firewalls drop the requests at the border; distributed firewalls do so at the host. A more interesting question is what is noticed by the host attempting to connect. Today, such packets are typically discarded, with no notification. A distributed firewall may choose to discard the packet, under the assumption that its legal peers know to use IPSec; alternatively, it may instead send back a response requesting that the connection be authenticated, which in turn gives notice of the existence of the host.

Firewalls built on pure packet filters cannot reject some "stealth scans" very well. One technique, for example, uses fragmented packets that can pass through unexamined because the port numbers aren't present in the first fragment. A distributed firewall will reassemble the packet and then reject it.

On balance, against this sort of threat the two firewall types are at least comparable.

Application-level Proxies

Some services require an application-level proxy. Conventional firewalls often have an edge here; the filtering code is complex and not generally available on host platforms. As noted, a hybrid technique can often be used to overcome this disadvantage.

In some cases, of course, application-level controls can avoid the problem entirely. If the security administrator can configure all Web browsers to reject ActiveX, there is no need to filter incoming HTML via a proxy.

In other cases, a suitably sophisticated IPSec implementation will suffice. For example, there may be no need to use a proxy that scans outbound FTP control messages for `PORT` commands, if the kernel will permit an application that has opened an outbound connection to receive inbound connections. This is more or less what such a proxy would do.

Denial-of-Service Attacks

It is impossible to lump all denial-of-service attacks into one basket. However, some statements can be made about particular known attacks.

The "smurf" attack (CERT Advisory CA-98.01, 5 January 1998) primarily consumes the bandwidth on the access line from an ISP to the target site. Neither form of firewall offers an effective defense. If one is willing to change the topology, both can be moderately effective. Conventional firewalls can be located at the ISP's POP, thus blocking the attack before it reaches the low-bandwidth access line. Distributed firewalls permit hosts to be connected via many different access lines, thus finessing the problem. For now, a distributed intrusion-detection system would be useful.

It may be possible to chew up CPU time by bombarding the IKE process with bogus security-association negotiation requests. Although this can affect conventional firewalls, inside machines would still be able to communicate. Distributed firewalls rely much more on IKE and hence are more susceptible. It is an open question whether a different key-exchange protocol will be needed to resist such attacks.

Conversely, any attack that consumes resources on conventional firewalls, such as many email attachments that must be scanned for viruses, can bog down such firewalls and affect all users. For that matter, too much legitimate traffic can overload a firewall. As noted, distributed firewalls do not suffer from this effect.

Intrusion Detection

Many firewalls detect attempted intrusions. If that functionality is to be provided by a distributed firewall, each individual host has to notice probes and forward them to some central location for processing and correlation.

The former problem is not hard; many hosts already log such attempts. One can make a good case that such detection should be done in any event. Collection is more problematic, especially at times of poor connectivity to the central site. There is also the risk of coordinated attacks in effect causing a denial-of-service attack against the central machine.

Our tentative conclusion is that intrusion detection is somewhat harder than with conventional firewalls. While more information can be gathered, using the same techniques on hosts protected by conventional firewalls would gather the same sort of data.

Insider Attacks

At first glance, the biggest weakness of distributed firewalls is their greater susceptibility to lack of cooperation by users. What happens if someone changes the policy files on their own?

Although there are technical measures that can be taken, as discussed earlier, these are limited in their ability to cope with serious misbehavior. That said, we assert that this problem is not a real differentiator. Even conventional firewalls are easily subverted by an uncooperative insider. SSH[13] can be used to tunnel TCP ports; external Web proxies such as `www.anonymizer.com` can bypass destination restrictions; end-to-end encryption can hide traffic, etc. In other words, an insider who wishes to violate firewall policy can do so with either type of firewall. As Marcus Ranum put it, "You can't solve social problems with software."

An insider who wishes to violate firewall policy can do so with either type of firewall.

On the other hand, distributed firewalls can reduce the threat of actual attacks by insiders, simply by making it easier to set up smaller groups of users. Thus, one can restrict access to a file server to only those users who need it, rather than let everyone inside the company pound on it.

It is also worth expending some effort to prevent casual subversion of policies. If policies are stored in a simple ASCII file, a user wishing, for example, to play a game could easily turn off protection. Requiring the would-be uncooperative user to go to more

IPSec can be used to implement stronger firewalls while eliminating many of the limitations of today's firewalls.

trouble is probably worthwhile, even if the mechanism is theoretically insufficient. For example, policies could be digitally signed, and verified by a frequently changing key in an awkward-to-replace location. For more stringent protections, the policy enforcement could be incorporated into a tamper-resistant network card.

A more serious issue concerns inadvertent errors when using application-level policies. An administrator might distribute configuration restrictions for Netscape Navigator or Microsoft's Internet Explorer. But what would happen if a user, in all innocence, were to install the Opera browser? The only real solution here is a standardized way to express application-level policies across all applications of a given type. We do not see that happening any time soon.

Conclusions

Firewalls are sometimes described as incompatible with the modern network environment. Furthermore, there are conflicts between a strong security technology – end-to-end IPSec – and firewalls, since the firewall can no longer examine the traffic. We have shown that the two are actually synergistic: IPSec can be used to implement stronger firewalls while eliminating many of the limitations of today's firewalls. We retain protection and centralized policy control; we eliminate dependency on topology, IP addresses, and the conflict with IPSec.

References

[1] M. Blaze, J. Feigenbaum, and A. Keromytis. "KeyNote: Trust management for public-key infrastructures." *Proceedings of the 1998 Cambridge Security Protocols International Workshop,* pp. 59–63. Springer, LNCS vol. 1550, 1999.

[2] Matt Blaze, Joan Feigenbaum, and Jack Lacy. "Decentralized trust management." *IEEE Symposium on Security and Privacy,* pp. 164–173, 1996.

[3] Matt Blaze, Joan Feigenbaum, and Martin Strauss. "Compliance checking in the PolicyMaker trust management system." *Proceedings of the 2nd Financial Crypto Conference,* Lecture Notes in Computer Science, 1998.

[4] Yair Bartal, Alain Mayer, Kobbi Nissim, and Avishai Wool. "Firmato: A novel firewall management toolkit." *Proceedings of the IEEE Computer Society Symposium on Security and Privacy,* 1999.

[5] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker.* Addison-Wesley, 1994.

[6] S. Kent and R. Atkinson. "Security architecture for the internet protocol." Request for Comments (Proposed Standard) 2401, Internet Engineering Task Force, November 1998.

[7] Andrew Koenig. "Automatic software distribution." *USENIX Summer 1984 Conference Proceedings,* pp. 312–322.

[8] William LeFebvre. "Restricting network access to system daemons under SunOS." *Proceedings of the Third USENIX UNIX Security Symposium,* 1992, pp. 93–103.

[9] Jeffrey C. Mogul. "Simple and flexible datagram access controls for UNIX-based gateways." *USENIX Summer 1989 Conference Proceedings,* pp. 203–221.

[10] David M. Martin, Sivaramarkrishnan Rajagopalan, and Aviel D. Rubin. "Blocking Java applets at the firewall." *Proceedings of the Symposium on Network and Distributed System Security,* San Diego, February 1997, pp. 16–26.

[11] J. Postel and J. Reynolds. "File transfer protocol." Request for Comments (Standard) STD 9, 959, Internet Engineering Task Force, October 1985. (Obsoletes RFC765; updated by RFC2228).

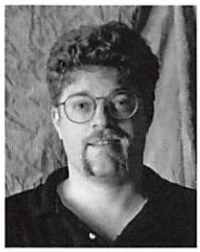
[12] Wietse Venema. "TCP WRAPPER: Network monitoring, access control and booby traps." *Proceedings of the Third USENIX UNIX Security Symposium,* 1992, pp. 85–92.

[13] Tatu Ylonen. "SSH – secure login connections over the Internet." *Proceedings of the Sixth USENIX Security Symposium,* 1996, pp. 37–42.

[14] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. "The KeyNote Trust-Management System Version 2," September 1999. RFC 2704.

selling security

Fear Leads to . . . the Dark Side



by Marcus J. Ranum

Marcus J. Ranum is CEO of Network Flight Recorder, Inc. <http://www.nfr.net>.

<mjr@nfr.net>

When I first started working with computer security, I used to wish sometimes, "If only people were more aware of the problem. Then I wouldn't have to start off by having to convince them they needed to do something – we could get right down to solving problems." Be careful what you wish for. It might come true. Today it seems that awareness of security is at an all-time high. But I'm afraid it's more because of hype and scare tactics than because of positive, useful awareness and education. In short, I think we're gaining in the "fear, uncertainty, and doubt" department and losing in the "clue" department. Is ignorance better than fear? As Yoda says, "Fear leads to anger, anger leads to hate, and hate leads to the Dark Side."

Computer security makes the news every day, or close to it. As I write this, the current brouhaha surrounds some undocumented and unexplained features of Microsoft's cryptography API: one of the variables – a public-key component – was named with a prefix of NSA. Is it a trapdoor in the crypto, installed by request of the National Security Agency, or is it something more innocent? Does the "NSA" prefix refer to the National Security Agency or to the "Next Security Attribute"? Well, we don't know yet. But speculation is flying, covering the complete range from "it's probably nothing" to "it's a plot spearheaded by the orbital mind-control lasers." The truth is out there. And it's probably banal, as truth often turns out to be. By the time you're reading this, we'll probably know more about what's really going on.

What's interesting to me about the NSAKEY incident isn't the existence of the key; it's the way in which the public was informed about the existence of the key. Basically, the information was dumped onto the Internet in the form of a conjecture – a perfect recipe for a tempest of conspiracy theories and wild speculation. I'm not trying to say that the person who discovered the variable did anything wrong, but I'd love to know why someone didn't dig deeper into the matter before going public with it. In the "hard sciences" we have textbook examples of what happens when you go public with a result before you've double-checked and done some quiet peer review: e.g., room temperature fusion.

So, what's really going on here? Are we trying to educate people about security, or are we hyping them with late-breaking news that hasn't been fully researched? I'm starting to lean toward the latter. I went to the Web site of the company that found the "NSA back door," read its FAQs on the issue, and browsed around a bit. The company consults on matters related to public-key infrastructures. The NSAKEY "almost certainly doesn't pose a threat to individual users." Oh, I see.

When "Joe User" reads about this stuff, he's going to conclude one of two things:

- So what? These security guys are always making mountains out of molehills.
- So what? Nothing is safe. So I'll just forget security.

What I fear is that, while trying to promote awareness of security, we're actually damaging our case by raising fear, uncertainty, and doubt over problems to which we can't offer solutions! Security guys are constantly amazed that, in spite of huge security problems, 99.9% of the world is comfortable with Windows. It's not that they're comfortable, guys, it's that we haven't offered them a viable, secure alternative.

Also, let's stop assuming "Joe User" is stupid. He understands vested interests. When a consulting firm specializing in public-key infrastructures does a press release about a problem with Microsoft's use of public keys, what do you think goes through his mind? A few years ago, when I worked at a company that built smartcard systems, we were overwhelmed with customer calls when someone announced a theoretical vulnerability in smartcard systems. About a week later, when the ruckus was dying down, we got a letter from the company that had publicized the vulnerability, offering us consulting services to see if our system needed fixing. I'm sure the timing was merely a coincidence.

I think what Yoda meant to say was, "Hype leads to fear, which leads to the search for a solution, which leads to apathy when no solution is available."

Tragically, many people overlook the obvious answer – which is that both sides are right!

Hyping Holes

There's also a responsibility issue. I know that responsibility is out of fashion, but humor me in my dotage. There's been a long and vigorous debate in the security community over the question of "full disclosure" of bugs. On one side, the argument is: "Disclosing the nature of security bugs makes us more vulnerable. Keep them secret and work quietly to fix them." On the other side, the argument runs: "Disclosing the bug forces the vendor to fix it in a timely manner. The bad guys already know about them, so let's publicize it." Tragically, many people overlook the obvious answer – which is that both sides are right! The devil's always in the details: how you disclose the bug, how you communicate with the vendor, and what you disclose versus what you keep secret.

I'm not going to try to carry on the disclosure debate, because that's not what this article is about. This article is about the wrong way (and, by negation, the right way) to sell security. The disclosure debate is irrelevant to selling security, but it's important because disclosure has become one of the marketing strategies of security practitioners.

Probably the best recent example I can think of is an IIS bug that was publicized by a small security company a number of months ago. The bug, which would be easy to implement from their description, suddenly pulled the rug out from under all the Web sites that were using the latest version of IIS. Microsoft rushed a fix out, and after a while the furor died down. I read a couple of news articles about the incident, and one part really stuck in my mind: the sequence of events. The flaw was found. Notice was sent to Microsoft. Microsoft did not release a patch and then stopped responding to email about the problem. So our intrepid heroes did what they had to do: they published the bug on the Internet. What's interesting about that? The whole sequence of events lasted *one week.* These guys actually expected a vendor that has to manage releases for skadjillions of products to push a patch out, what, overnight? And they were worried that Microsoft had stopped the dialog with them? It was probably the *weekend!* When the bug was publicized, exploits were available for it within three hours of its posting, and innocent Web-site administrators' days were being ruined shortly thereafter. Not to be outdone, within six hours of the bug being publicized, the discoverers released their own exploit tool.

Now, either these guys live entirely in Internet Time (where one hour equals a day) or they had an agenda beyond just helping Microsoft fix a vulnerability in its software. I suspect the latter. Frankly, I think they did it to market themselves. I doubt there was *any* speed at which Microsoft could have responded that would have been adequate. What they wanted wasn't security; they wanted attention, Web hits, and, by extension,

Sure, the guy will help you secure your Web site; he honed his skills by making your fellow sysadmins' lives hell.

money. Did I mention that they sell vulnerability-assessment tools and other security products?

People, this is not how to sell security. What this does is convince everyone that achieving security is a hopeless task, doomed to dismal failure. Possibly it teaches everyone to be extremely skeptical about security alerts, because they're all clearly a bunch of hype, just like that Y2K thing we used to hear about.

Aiding and Abetting

There's a company that sells books on how to make bombs and drugs and how to commit identity fraud, "for educational purposes only." Now, I'm a staunch defender of the Bill of Rights – even the unpopular parts of it – but I happen to think that "rights" do not absolve one from "responsibilities." One of the other ways security is being marketed is by demonstrations of security-breaking skill or know-how. The logic goes like this: "I know 300 ways to break into systems. Therefore I know how to protect yours." The guy who knows 600 ways to break in must be twice as skillful, I guess.

This has started a trend that I believe makes the problem worse rather than better. Instead of just having the "bad guys" trying to find and exploit holes in systems, now we have the "good guys" doing it too, or hiring "ex-bad guys," repackaging them as "good guys," and selling them to you for $400 an hour. Sure, the guy will help you secure your Web site; he honed his skills by making your fellow sysadmins' lives hell. No apologies necessary, it's just business, isn't it? There are security companies that brag about their SWAT teams, which consist of largely the same guys you're trying to protect yourselves against. They're not "bad guys" anymore, though. Now they've got stock options and drive Ferraris.

A plethora of sites distribute attack tools "for educational purposes." These sites are indirectly responsible for the rapidly growing population of script-kiddies who have learned how to use point-and-hack attack tools. Some of these sites preface their tools collections with mealy-mouthed self-justification about how it's a useful meeting ground between security professionals and the "other half." Thanks, guys.

What kind of signals does all this send to our young script-kiddie who is looking at starting a career breaking into systems? Develop exploits, publish them, and hype yourself. . . . It's okay to do it; everyone else is. . . . When you get tired of being on the Dark Side, join the Jedi.

It hurts reputable security professionals as well. I was speaking at a conference the other day and made a comment about how I don't know how to break into systems. Several people in the room giggled. But it's true – I don't. Why does everyone assume that in order to be a good bank guard you have to have knocked over a few liquor stores first? My hacking skills aren't even as good as the lamest script-kiddie's, yet I get two or three emails a month from hotmail.com addresses asking me if I know any secret ways through XYZ brand firewall. If I knew of any, I'd have contacted the vendor by now, and given them more than a week to fix it, too.

Let's Get Real

Since I've complained about people who create problems but don't offer solutions, I guess I need to make some suggestions about how to roll back the tide of security hype and fear. It's not easy.

First, I offer the "show me the money" litmus test for hype:

- If you see a press release about a security problem, it's hype.

Nobody is going to pay a newswire service good money to announce a bug unless they expect to get something out of it in return. When you read about a shocking new vulnerability found in something, research not only the vulnerability but the individual or organization that announced it. Ask yourself if they happen to sell a solution to the problem, and keep your skepticism in gear.

Second, I offer the "be part of the solution" test:

- If the announcement/product/exploit makes things worse for people, it's part of the problem and is therefore hype. If it makes things better, then it's good.

When someone releases a piece of software, does it improve the security of your system or erode the security of someone else's? Avoid the latter, promote the former. Distributing cracking tools "for educational purposes" is an insult when you consider the number of great, free, beneficial tools that are available. Tools like `tcpwrappers`, `postfix`, COPS, `ipfilt`, and others serve to protect – their authors deserve the laurels.

Tools like `tcpwrappers`, `postfix`, *COPS,* `ipfilt`, *and others serve to protect – their authors deserve the laurels.*

Last, I urge you to disseminate positive information, not system-cracking techniques. Why waste the time writing an article on how to stack-smash sendmail, when you could do code reviews and help the authors fix their bugs? Why write tools to break into firewalls when you can write tutorials on how to improve them? Oddly, you'll make more friends in the system-administrator community if you help them solve problems than you will if you keep shoving problems in their faces. If it's money you want, you'll make more money selling solutions than creating problems. If your solution is worth buying, it will be obvious enough that you don't need to release an exploit script to force your customers to buy your product out of fear.

Obviously, I'm opinionated on these matters, and I don't expect everyone will agree with me. In either case, if you work with security, I urge you to denounce hype when you see it. As computers and networks continue to infiltrate our lives, security is going to be a bigger and bigger problem. There's no sense making it bigger still by selling fear.

full disclosure

The Future of Vulnerability Disclosure?

by Jeremy Rauch

Jeremy Rauch is a founder of SecurityFocus.com, a Web site that disseminates security information to the public and maintains a number of popular forums and mailing lists, including the Incidents and Bugtraq mailing lists.

<jrauch@securityfocus.com>

Every day, flaws are discovered in software. It's inherent in computer technology that as computers become more and more powerful, and the operating systems and applications become more and more complex and feature-rich, problems are going to creep in. In a lot of cases, these bugs are little more than the nuisances the term "bug" implies. Sometimes, however, the problems are of a less benign nature and have security implications. Flaws – ranging from those that prevent a user from reading email, to those that might allow someone to read millions of people's email messages – can and do happen.

When problems arise, different kinds of actions can result. Often, the person finding the problem will contact the vendor who wrote the product. Other times, the flaw will be discovered because it was used to compromise the security of a machine. Some companies have whole groups of people whose sole mission is to hunt out vulnerabilities in software and work with the vendor in remedying them. In the last five years or so, however, the growth of people participating in what has been dubbed "full disclosure" has skyrocketed. Its critics are as numerous as its champions, and all the commotion often makes it hard to determine just what full disclosure really is and why it works where other methods fail.

The goals of this article are to describe full disclosure, expose its positive aspects, compare and contrast it with more "mainstream" methods, and try to get at the heart of the controversy. I hope the facts about full disclosure, separated from the emotion its supporters and detractors bring to it, will speak for themselves.

What's It All About?

Full disclosure has really come into its own in the last five or so years with the explosive popularity of mailing lists like Bugtraq. Started in 1993, Bugtraq was designed from the start as an open forum for the no-holds-barred discussion of security vulnerabilities. The underlying theory behind Bugtraq and the other full-disclosure mailing lists that followed it was that the only way to improve security is to understand the problems that exist. No more would the details of security problems be limited to closed mailing lists of so-called security experts or detailed in long, overwrought papers from academia. Instead, the information would be made available to the masses to do with as they saw fit.

Naturally, a backlash occurred that has continued to this day. The naysayers argued from the start that full-disclosure mailing lists such as Bugtraq did more harm than good. Arming the hacker "enemy" with details previously made available only to those who were deemed worthy would bring chaos, a security Armageddon. Six years later, and with no collapse of the Internet, there are still those who don't fully appreciate the goals of full disclosure, and very few who understand the impact that full disclosure has, and will have, on security in the future.

Full disclosure, in a nutshell, means disseminating information about security vulnerabilities. It is not about any one aspect; it is not publishing specific vulnerability exploits, nor is it about embarrassing vendors. Its sole purpose is to arm the security-conscious with the knowledge necessary to evaluate risks and take applicable action. This may mean using the vulnerabilities made public in order to audit a network for those vulnerabilities. It might also mean simply downloading patches and installing them. Full

disclosure endeavors to give people the flexibility to take what they feel is appropriate action, rather than be hampered by insufficient information.

Full disclosure is in many ways akin to the open-source movement that's taking the computer world by storm. Open source allows for peer review, learning, and collaboration that leads to making better software. Full disclosure has similar goals. By making the details of vulnerabilities public, it seeks to educate and inform, and at the same time to provide a basis upon which to take further action.

For example, buffer overruns were once an obscure topic, but now they have been discussed and dissected to the point where many programmers understand how to prevent them, even if they are incapable of writing an exploit. Where there were once ten new exploits based on the buffer-overrun concept each week, the rate at which they are found has slowed to a trickle. The discussion of these problems in an open, collaborative forum helped to promote understanding, which in turn has significantly reduced the number of vulnerabilities of this type. There are other examples of this –from race conditions to file-permission problems to authentication mechanisms and even simple things like password management – that are now understood by enough people that they no longer plague every other program in existence. In the same way that an open-source project benefits from the input of programmers worldwide scrutinizing its code base, system security has benefited from the scrutiny of full disclosure.

The Critics

Critics of full disclosure argue that the risks associated with publishing information outstrip its benefits. Overworked administrators and security people are incapable of keeping up, they say, and urge that the best route to take is through the vendor, software developer, or one of the multitude of incident-response teams that exist. They take care of fixing the problem and limit the information given out, making it difficult to create an exploit from the information.

This "quiet" approach has two major flaws that have been seen time and time again, however. The first flaw is the assumption that the honest person who reports the vulnerability to a vendor is the first one to uncover it. Exploits often exist in the hands of the so-called "enemy" for months, sometimes even years, before the vendor is notified. The second flaw is that in a number of instances a vulnerability has been reported to a vendor, and in the time between notification and patching, exploits for this vulnerability have floated to the surface. This doesn't necessarily mean that information about security flaws finds its way from the inside vendors into the wild; it can also indicate that for any problem an individual finds, chances are that someone else has already found it or will find it. In the eyes of the full-disclosure advocate, it's better to know about a problem and be unable to obtain a vendor patch than it is to be left exposed, regardless of how tightly guarded a secret the vulnerability is thought to be.

Shooting the Messenger

Vulnerabilities exist whether or not they are found and whether or not they are reported to a vendor or on a mailing list like Bugtraq. This is a fact lost on many people. When software is written, if proper care isn't given to making sure just the proper number of bytes can be copied or that a file being created by root isn't really a symbolic link, a vulnerability exists. It may be found instantly, or it may be found ten years later. It may be quietly patched, or it may make the front page of the newspaper. Often people blame the person who discovers a problem for doing something wrong, when in fact that person was little more than a messenger. Only by identifying the problem can programmers learn from their mistakes and future problems be limited.

In the eyes of the full-disclosure advocate, it's better to know about a problem and be unable to obtain a vendor patch than it is to be left exposed.

Because full disclosure is such a flexible and broad approach to security issues, people are often confused about what the delivery mechanism usually is and just who is doing the disclosing. The most consistently active people in the full-disclosure community are not hackers, as some critics might suggest, but people from a wide variety of backgrounds. System administrators, academics, security consultants, developers, and general security enthusiasts all participate. The information disclosed may be a short note from an individual who suspects that his or her machine was compromised by a given service. It may also be a multi-page dissection of a problem.

Exploits for the problem are often included, but even more common are workarounds and patches developed by individuals as stopgap measures until official patches are available. Contrary to popular belief, vendor notification and full disclosure are not mutually exclusive. Many people choose to notify vendors prior to disclosing information and give details to the public only after the vendor has had an opportunity to address the problem. Since the real goal is to improve security, it is rare for people to post a vulnerability without giving thought to its impact. Posters to the Bugtraq mailing list are always encouraged to notify vendors before sending details to the mailing list.

Full disclosure has its flaws. As many have pointed out, full disclosure does put powerful, dangerous code into the hands of some who may not handle it responsibly. There is always potential for an exploit that is made public to cause major damage. The question that's difficult to answer, however, is: What would the impact of the problem have been had it not been disclosed? Fewer hosts might have been compromised, but whether it would have been noticed is questionable, and the problem might have continued to be a problem. Recently, a vulnerability and exploit for the IIS Web server was released to the public. Millions of hosts were vulnerable, yet relatively few hosts were attacked after the problem was detailed. Patches were quickly issued, administrators were made aware of the problem, and the Web continued to grow without the problems some might have predicted.

Information about subscribing to Bugtraq and a variety of other mailing lists related to full disclosure and security can be found at <http://www.securityfocus.com>.

Many also point out that often problems that are not being exploited by hackers are made public, as are the means to exploit these problems. It's very difficult to determine just what is being actively exploited in the wild. Simply pointing out that dozens of Web sites are not being defaced is insufficient. In addition, it is often simply a matter of time before these vulnerabilities are found by other parties whose intention may be to keep the vulnerabilities to themselves for nefarious purposes. Vulnerabilities disclosed only to the software author or vendor often seem to rise slowly to the surface in the time between their notification and the patching of the problem; this seriously calls into question whether it's possible to keep a vulnerability secret for long.

With the explosive growth of the Internet, it becomes more and more important to be able to keep abreast of security developments and problems. Full-disclosure mailing lists and newsgroups fill this role perfectly. Vulnerabilities and solutions are disclosed and discussed, and the reader can walk away with enough information to feel comfortable with taking action for the sake of security. The days when one could ignore full-disclosure forums out of some mistaken notion of security integrity are gone. The information being talked about on these lists is the same information being traded among the underground elements looking to break into machines. Only by being well informed can one hope to be secure.

welcome to the big city

Incident Reporting Helps the CERT™ Coordination Center Keep Pace with a Rapidly Expanding Internet

The Internet has been around in some form for the past 30 years. Security has always been a concern but has historically been more of an afterthought than a requirement. The history of Internet security can be compared to life in a town. When the town is small, people are likely to know and trust one another, doors are left unlocked, and generally there is little or no crime. However, as the town grows, security and crime become greater concerns.

by Jeffrey J. Carpenter

Jeffrey J. Carpenter is a senior Internet security technologist at the CERT Coordination Center (CERT/CC) in the Software Engineering Institute at Carnegie Mellon University,

<jjc@cert.org>

The Internet of today has grown to something in orders of magnitude greater than the largest of metropolitan cities. Consequently, there is an increase in crime and concern about security. Moreover, dependency on the Internet as a communications infrastructure has increased. At the same time, however, intruder technology has become significantly more sophisticated, and the expertise required to be a successful intruder has decreased. Fortunately, security awareness among system administrators and Internet users has increased, but much work remains.

The CERT Coordination Center Is Born

The Internet was still very much a small city in November 1988, when a Cornell University graduate student let loose the notorious Internet worm that brought down much of the Internet and demonstrated the growing network's susceptibility to attack. Once a group of researchers drawn from government and the academic community successfully contained the worm, the National Computer Security Center (part of the National Security Agency) initiated a series of meetings to discuss how to prevent and respond to such occurrences in the future.

Shortly thereafter, the Defense Advanced Research Projects Agency (DARPA) announced its intention to fund development of the CERT Coordination Center (CERT/CC). DARPA chose the Software Engineering Institute (SEI) on the campus of Carnegie Mellon University, in Pittsburgh, Pennsylvania, as the new center's home. The SEI was charged with establishing the capability to quickly and effectively coordinate communication among experts during security emergencies in order to prevent future incidents, and with building awareness of security issues in the Internet community at large.

Since its inception in 1988, the CERT/CC has responded to more than 17,800 security incidents that have affected over 235,000 government, academic, and corporate sites. Consequently, the time required to resolve computer security incidents and repair computer-system vulnerabilities has decreased. The resulting incident-response and security-improvement practices developed have led to networked computing systems that are more resistant to attack and less likely to be compromised.

The Incident Reporting Process

Input from the community is critically important to the CERT/CC. The community provides us with the necessary raw data in the form of incident reports, vulnerability reports, alerts from intrusion-detection systems on networks, and discussions with other response teams and experts. Public mailing lists and security Web sites are also monitored.

The majority of this input data arrives in the form of incident reports from system and network administrators. An incident report is a collection of data that has been identi-

Incident Notes describe current intruder activities that have been reported to the CERT/CC incident-response team.

fied by someone as an attack. The format and transport for an incident report is typically unstructured text sent by email to <cert@cert.org>, or by a telephone call to the CERT Hotline (412 268-7090).

Since the CERT/CC was established, its goals have been identifying Internet security trends, detecting attacks spanning multiple administrative domains, and handling attacks targeted against or affecting the Internet infrastructure. The mechanisms used to accomplish these goals have continued to be developed and refined over time. Data from multiple sources is processed in an attempt to positively identify attacks targeted at, or affecting significant portions of, the Internet infrastructure.

When an incident is reported to the CERT/CC, an analyst works to determine its priority. This involves interpreting, extracting data from, analyzing, and recording information on the basis of the incident report. Part of the analysis that occurs with a new incident report is correlation with data that has been received from all input sources and past incidents. Collections of data representing related attacks are referred to as an incident. One incident can represent something minor, such as a single probe to a single site, or something significant, such as the Melissa virus. Additional analysis involves identifying instances of known methods or signatures of attack. Any attack method identified that cannot be represented by a common, well-known signature is investigated by the incident analyst until the details of the attack can be determined. When novel methods of attack are uncovered, the analyst records a signature for that attack and determines what the likelihood of attack is and what the threat is for the use of the attack on the Internet infrastructure. If the threat exceeds a certain threshold, a CERT Advisory is issued.

On the basis of the information received from the Internet community, critical information about specific threats is disseminated through security alerts, such as CERT Advisories, Incident Notes, Vulnerability Notes, and Vendor-Initiated Bulletins. CERT Advisories address Internet security problems. They offer an explanation of the problem, information that helps determine if a site has the problem, fixes or workarounds, and vendor information. Among the criteria for developing an advisory are the urgency of the problem, potential impact of intruder exploitation, and the existence of a software patch or workaround. CERT Summaries are published as part of our ongoing efforts to disseminate timely information about Internet security issues. The summary is typically published four to six times a year. The primary purpose of the summary is to call attention to the types of attacks being reported to us.

We also publish two Web documents, Incident Notes and Vulnerability Notes, as an informal means for giving the Internet community timely information relating to the security of its sites. Incident Notes describe current intruder activities that have been reported to the CERT/CC incident-response team (<http://www.cert.org/currnet/>). Vulnerability Notes describe weaknesses in Internet-related systems that could be exploited but that do not meet the criteria for advisories.

Lessons learned from incident handling and vulnerability analysis are made available to users of the Internet through a Web site and FTP archive of security information and products. These include answers to frequently asked questions, a security checklist, tech tips for system administrators, security tools such as TCP wrappers, research and technical reports, and a handbook for new computer security incident-response teams (CSIRTs). Members of the Internet community can subscribe to receive advisories by email. Subscription information is available on the CERT Web site: <http://www.cert.org>. At present there are more than 100,000 addresses on the public mail subscriber listing

with many of those as mail exploders (a higher-level address that forwards mailings to its own individual subscribers). As a result, it is estimated that CERT mailings reach over half a million addresses.

The most up-to-date information about ongoing attacks can be found on the Current Activity section of our Web site. Other outputs include educational documents and vulnerability alerts to affected vendors.

The Benefits of Incident Reporting

Ultimately, incidents reported to the CERT/CC benefit all parts of the Internet community. In more severe cases, our staff may provide direct assistance in resolving an incident. A minor or seemingly insignificant piece of incident data at a reporting site may represent part of a much larger and more significant attack affecting multiple sites. Receiving data from multiple sources helps us to have a more accurate understanding of the current state of the Internet. This information is of fundamental importance for detecting attacks affecting significant parts of the Internet infrastructure and preventing the spread of such attacks by producing Advisories. We receive many valuable reports from sites that need no specific assistance. These sites let us know about the activity they are seeing or when they see new vulnerabilities or types of attacks. We encourage sites to report information to us even if they do not need assistance, because incident reporting adds to the body of attack knowledge that will find its way back to the community in the form of our advisories and other documents.

In addition to helping detect and scope the significance of attacks, incident reports provide a basis to determine trends and statistics in Internet security. This type of information is compiled regularly in our publications and frequently discussed in public forums such as conferences.

From Reactive to Proactive

Over the years, the CERT/CC has evolved from a handful of technical staff reacting to computer security incidents to a multidisciplinary team of professionals working to prevent future incidents as well as respond to them.

Based on this experience, our research agenda has been structured around several major components, including security evaluation and improvement, simulation of interconnected systems and infrastructures, assisted and automated detection and response techniques, and advanced vulnerability analysis.

In the area of automated incident-detection and response, we continue to work toward automation of incident reporting and automated incident data sharing. Tools are also under development within the program to support the automated upgrade and patching of large heterogeneous networks. In conjunction with the result of vulnerability handling and prioritization, these automated upgrade tools can assist in the protection of large distributed networks such as those found in railway systems, power distribution, and telecommunications.

Keeping Pace with the Threat

The Internet is an environment in which intruders form a well-connected community and use network services to distribute information quickly on how to maliciously exploit vulnerabilities in systems. Intruders dedicate time to developing programs that exploit vulnerabilities and to sharing information. They have their own publications, and they regularly hold conferences that deal specifically with tools and techniques for defeating security measures in networked computer systems.

In more severe cases, our staff may provide direct assistance in resolving an incident.

In contrast, the legitimate, often overworked system administrators frequently find it difficult to take the time and energy from their normal activities to stay current with security and vulnerability information, much less design patches, workarounds, tools, policies, and procedures to protect the computer systems they administer.

In helping the Internet community work together, the CERT/CC and other incident-response teams face policy and management issues that are perhaps even more difficult than the technical issues. Most important, the Internet community needs to work together closely to keep pace with an emerging threat and to ensure that future products and services are able to survive.

After reading Jeff's article, Rik Farrow asked if he would explain when CERT/CC members would actually participate in handling an incident. Jeff's response:

We have to use our resources to have the most effect on security across the Internet. One-on-one incident response is very expensive and helps an ever smaller percentage of the Internet community. The number of incidents reported to us is increasing geometrically. We now receive more than 30 incidents per day. We don't have the resources to handle each incident personally. Even if we did, I don't think that handling would have the same impact that it did in the early days. We have much more impact by taking the information we see from incident reports and providing that back to the Internet community so they can take steps to prevent themselves from being victimized. We need to work on improving the volume and organization of the information we provide to the community in order for us to continue to be a valuable resource for system and network administrators.

There are certain types of incidents that receive priority and will receive a response, such as cases where someone's life is in danger or the infrastructure of the Internet is in danger. After those kinds of incidents, we want to focus on incidents where we learn something new, something we will be able to use to provide information to the Internet community.

USENIX The Advanced Computing Systems Association

MEMBERSHIP INFORMATION

Indicate the category of membership which you prefer and send appropriate annual dues

- ☐ ***Individual:*** *$ 80*
- ☐ ***Full-time Student:*** *$ 25*
  Please enclose a copy of your current student ID.
- ☐ ***Educational:*** *$ 200*
- ☐ ***Corporate:*** *$ 400*
- ☐ *Supporting – USENIX –* ☐ *$ 1000* ☐ *$ 2500* *SAGE –* ☐ *$ 1000*

A designated representative for each Educational, Corporate, and Supporting membership receives all USENIX conference and symposia proceedings published during their membership term plus all member services. Supporting members also receive one free full page ad in *;login:*, more member-rate discounts for technical sessions, a half-price rental of the mailing list, 10% discounts on vendor displays booth rentals, and a link to their URL from the USENIX Web site.

$50 of your annual membership dues is for a one-year subscription to *;login:*

SAGE, the System Administrators Guild

SAGE, a Special Technical Group within the USENIX Association, is dedicated to the recognition and advancement of system administration as a profession. To join SAGE, you must be a member of USENIX.

☐ *Individual: $30* ☐ *Students: $15*

MEMBERSHIP APPLICATION: ☐ *New* ☐ *Renewal*

Name _____

Address _____

City: _____ State: _____ Zip: _____ Country: _____

Phone: _____ Fax: _____ Email: _____

MEMBER PROFILE: Please help us serve you better!

By answering the following questions, you provide us with information that will allow us to plan our activities to meet your needs. All information is entirely confidential.

What is your job function?

1. ☐ System/Network Administrator
2. ☐ Consultant
3. ☐ Academic/Researcher
4. ☐ Developer/Programmer/Architect
5. ☐ System Engineer
6. ☐ Technical Manager
7. ☐ Student
8. ☐ Webmaster
9. ☐ Security

What is your role in the purchase decision

1. ☐ Final
2. ☐ Specify
3. ☐ Recommend
4. ☐ Influence
5. ☐ No role

☐ I do not want USENIX to email me notices of Association activities.

☐ I do not want my address made available for commercial mailings

PAYMENT OPTIONS:

Total enclosed $ _____

☐ I enclose a check/money order made payable to USENIX Association

☐ Enclosed is our purchase order (Educational, Corporate, and Supporting members only, please).

☐ Charge my: ☐ Visa ☐ MasterCard ☐ American Express

Account # _____ Expiration Date _____

Name on Card _____

Signature _____

Outside the USA? Please make your payment in US dollars by check drawn on US Bank, Visa/MasterCard, International postal money order

USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, California 94710 ■ Phone: 510 528 8649 ■ FAX: 510 548 5738 ■ Email: office@usenix.org

revised 9/9/99

Announcement and Call for Papers **USENIX**

3rd Large Installation System Administration of Windows NT / 2000 Conference

Sponsored by USENIX, the Advanced Computing Systems Association, and by SAGE, the System Administrators Guild

http://www.usenix.org/events/lisa-nt2000

July 30 - August 2, 2000 **Seattle, Washington, USA**

Important Dates

Submission proposals due: *Feb. 16, 2000*
Notification to authors: *March 13, 2000*
Final papers due: *June 1, 2000*

Conference Organizers

Conference Co-Chairs

Phil Cox, *SystemExperts*
Jesper M. Johansson, *Boston University*

Program Committee

Kip A. Boyle, *SRI Consulting*
Russ Cooper, *RC Consulting*
Alan Epps, *Intel*
Aeleen Frisch, *Exponential Consulting*
John Holmwood, *NOVA Gas Company*
David LeBlanc, *Microsoft Corp.*
Todd Needham, *Microsoft Corp.*
Jason Reed, *System Experts*
Dave Roth, *Microsoft Corp.*
Martin Sjölin, *Warburg Dillon Read*

Overview

By late next summer, we will have all undergone the flurry or flutter of Y2K, and many of us will be in the midst of another similar dilemma: migrating to Windows 2000. Sites around the world are all slowly beginning to plan this migration, some sooner than others, and all are seeking answers from those who have blazed the trail ahead of them. The Large Installation System Administration of Windows NT conference, LISA-NT, is a forum to bring system administration professionals together to discuss workable solutions to the issues of administering and scaling all versions of the NT environment. In 2000 that will mean migration issues for many of us, and just keeping our heads above water with Windows NT 4.0 for others.

LISA-NT 2000 will bring together peers and experts in our field to discuss leading edge solutions that have a proven track record of working. LISA-NT is put together by and for Windows NT administrators who need solutions to problems such as integration, migration, security, and management using today's technology. We invite you to submit technical papers as well as proposals for invited talks, panel sessions, tutorials, and work-in-progress reports. There are also opportunities for Birds-of-a-Feather sessions and demonstrations of products and solutions. Please review this call for papers, prepare a submission, and join us in making LISA-NT 2000 the premiere conference for system administrators of distributed NT-based environments.

This conference will last four days. Two days of tutorials will be followed by two days of technical sessions including refereed papers, invited talks, and works-in-progress. On August 3-4, 2000, the Fourth USENIX Windows Operating Systems Symposium will be held in the same location.

Topics

LISA-NT is about creating a community of system administration professionals, which come from diverse computing environments, to give them an opportunity to discuss problems and share solutions. Submissions may address administration methodologies (i.e. "This is what I do and why") as well as implemented solutions (e.g. "This is the tool I wrote in Perl or VBscript or any other language" or "This is how I turned my Master-Domain model into a functional Active Directory"). To facilitate these discussions, we encourage you to consider these topics:

Management & Migration

- Management of homogeneous Windows NT networks
- Management of NT in heterogeneous environments
- Large-scale or distributed NT management solutions and experiences
- Locally developed administration tools and techniques
- Models for centralized, remote or distributed management
- Reduction of the total cost of ownership for users and/or workstations
- Backup and restore methodologies
- User account management
- Migrating existing domain structures to Windows 2000 Active Directory
- Large-scale configuration, roll-out, and maintenance of NT workstation and application software
- Management of NT for heterogeneous user constituencies
- Maintaining or enhancing NT security posture during migration to Windows 2000

Sharing and Integration

- Integration of NT into complex, existing environments
- Integration of NT (workstation and/or server) into existing UNIX or Netware environments
- Integration of Unix into NT environments
- Integration of an existing Unix Kerberos Realm with Windows 2000 Active Directory
- Integration of Windows 2000 and Windows NT 4.0 security functionality
- Use of cross-platform distributed services
- Remote access
- Laptops and PDA's running and/or accessing NT
- Deployment of Windows 2000 and/or Windows NT 4.0 in extranets
- Use of IPsec and other VPN solutions in homogeneous or heterogeneous NT/Win2K environments
- Strategies for deployment of Windows 2000 into heterogeneous networks

Tools, Experiences, and Issues

- Successful use of third-party applications during daily administration
- Surveys of examined tools (good and bad) while implementing a solution
- Performance tuning and measurement
- E-mail management

- Central name service and browsing
- Software licensing, deployment, and upgrade management
- Security issues, including policies, education, response, and fixes
- The registry

Best Paper Awards

Awards will be given for the best paper and best student paper at the conference.

What to Submit

The program committee seeks submissions from the Windows NT system administration community in the following formats:

Technical White Papers

We seek papers relating work of general interest to system administrators of Windows NT, particularly technical papers that reflect hands-on experience or describe implemented or attainable solutions. Submissions will be judged on the quality of the written submission and whether or not the work advances the art and science of NT system administration.

A paper submission should:

- Contain a short abstract
- Include an outline of the paper. If you have a completed paper, you may submit it instead of the abstract and outline.
- Conform to the "How and Where..." instructions below

Please see the detailed author instructions, which include a sample abstract, for more information: *http://www.usenix.org/events/lisa-nt2000/cfp/guidelines.html*

Authors of an accepted paper must provide a final paper for publication in the conference proceedings. At least one author of each accepted paper will present the paper during the technical track of the conference. These presentations generally include a 20-minute talk with 5-10 minutes of questions from the audience.

Conference proceedings containing all full technical white papers will be distributed to attendees and, following the conference, will be available online to USENIX members and for purchase.

The LISA-NT Conference, like most conferences and journals, requires that papers not be submitted simultaneously to another conference or publication and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Invited Talks/Panel Sessions

These survey-style talks given by experts range over many interesting and timely topics. Invited talks fit in a 90-minute session with a short question and answer session at the end. Please submit a proposal for an invited talk or a panel session to: *lisantpapers@usenix.org*

Work-in-Progress Reports (WIPs)

Work-in-Progress Reports (WIPs) are short talks that introduce new or ongoing work. If you have work you would like to share or an interesting idea that is not quite ready to be published as a paper, a WIP is for you. Acceptance will be based upon the applicability and scalability of the proposed solution. To submit a WIP, email a description of the problem and your (possibly incomplete) solution if necessary, and why the problem you are addressing is interesting, to: *lisantpapers@usenix.org*

Tutorials

On July 30 and 31, there will be full and half-day tutorials in all areas and levels of expertise for Windows NT system administrators. Previous tutorial sessions have covered topics such as "Windows NT Security," "Windows NT Internals," "Configuring Samba, Avoiding Common Pitfalls," and "Administering Windows NT DHCP and DNS servers."

If you are interested in presenting a tutorial at LISA-NT, please contact the USENIX tutorial coordinator:

Daniel V. Klein
Email : *dvk@usenix.org*
Phone : +1.412.422.0285
Fax : +1.412.421.2332

How and Where to Send Submissions

Please email your submission to *lisantpapers@usenix.org* in one of the following formats (listed in order of preference). If you enclose files as an attachment to your submission, please use MIME encoding.

- RTF
- Microsoft Word 97
- Postscript formatted for 8.5" x 11" page
- HTML
- Plain text with no extra markup

A cover letter with the following required information must be included with all submissions:

Authors : Names and affiliation of all authors, and an indication of which, if any, are full-time students
Contact: Primary contact for the submission
Address: Contact's full postal address
Phone: Contact's telephone number
Fax: Contact's fax number
Email: Contact's e-mail address
URL: For all speakers/authors (if available)
Category: Category of the submission (paper, invited talk, panel, WIP, BoF, demo/poster session)
Title: Title of the submission
Needs: Audio-visual requirements for presentation

We will acknowledge receipt of a submission by email within one week.

Registration Materials

Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available in May 2000. If you wish to receive the registration materials, please visit the symposium Web site or contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: *conference@usenix.org*

Announcement and Call for Papers **USENIX**

4th USENIX Windows Systems Symposium

(Formerly the USENIX Windows NT Symposium)

http://www.usenix.org/events/usenix-win2000

August 3-4, 2000 **Madison Renaissance Hotel, Seattle, Washington, USA**

Important Dates

Technical paper submissions due: *February 11, 2000*
Notification of acceptance: *March 13, 2000*
Camera-ready final papers due: *May 18, 2000*

Symposium Organizers

Symposium Co-Chairs

J. Bradley Chen, *Appliant Inc.*
Richard Draves, *Microsoft Research*

Symposium Steering Committee

Andrew Hume, *AT&T Labs Research*
Michael B. Jones, *Microsoft Research*
Werner Vogels, *Cornell University*

Program Committee

Ed Felten, *Princeton University*
Jim Gray, *Microsoft Research*
Brad Myers, *Carnegie Mellon University*
Karin Petersen, *Xerox Palo Alto Research Center*
Mendel Rosenblum, *Stanford University*
Dave Steere, *Oregon Graduate Institute*
Werner Vogels, *Cornell University*
Bill Weihl, *Compaq Systems Research Center*
Rumi Zahir, *Intel Corporation*

Overview

Following the tradition established by three very successful symposia, the Fourth USENIX Windows Systems Symposium will provide a forum for the discussion of research and advanced engineering use of the Windows platform. The symposium brings together professionals from academic and industrial backgrounds who are actively using Windows platforms (whether NT, 9x, or CE) and want to discuss ideas and share information, experiences, and results. The symposium will include technical presentations of refereed papers, invited talks, a Work-in-Progress session, informal Demo/Poster sessions, and Birds-of-a-Feather sessions. The primary purpose of the symposium is to facilitate two days of useful interaction among the participants.

On August 1-2, 2000, the Third Large Installation System Administration of Windows NT/2000 Conference (LISA-NT 2000) will be held in the same location. Two days of tutorials will precede both conferences.

Topics

Papers that present research results, analyze problem areas, draw important conclusions from practical experience, or facilitate discussion are especially welcome. We particularly encourage submissions that address key issues for mission-critical, real world system deployments, including:

- **Availability**: Experience with and research into deploying mission critical applications on Windows-based infrastructure.
- **Scalability**: Pushing the Windows platform to the limit. What is the limit? How can it be extended?
- **Manageability**: The use of various models, strategies and tools to address large-scale Windows or mixed networks, and large clusters of Windows machines.
- **Ubiquity**: Serving the needs of a broad range of devices, including mobile and embedded systems.

Papers that provide analysis or share experience on the following topics are also encouraged:

- **Applications**: Development and deployment of large applications environments.
- **User Interfaces**: Ideas and techniques for enhancing usability and improving the user experience.
- **Programming Environments**: Programming environments to exploit the wealth of Windows functionality.
- **Tools and Utilities**: How to make Windows a highly productive system.
- **Security**: Extending the Windows security model, and using Windows in highly secure environments.
- **Integration**: Architectural challenges such as heterogeneity and scale encountered in real world deployments.

What to Submit

We are soliciting technical papers on concepts, research, and experiences relevant to researchers using Windows. We particularly encourage researchers who have achieved major results using Windows to present them at the symposium. Even if the result itself has been reported elsewhere, a paper that explicitly addresses the benefits, challenges, and interesting systems results specific to Windows platforms is an appropriate submission for this symposium. (Of course, verbatim resubmission of previously published manuscripts is not acceptable.)

Authors of an accepted technical paper must provide a final camera-ready paper for publication in the symposium proceedings. Symposium proceedings containing all full technical papers will be distributed to attendees and, following the symposium, will be available online to USENIX members and for purchase.

Note that USENIX conferences, like most conferences and journals, require that papers not be submitted simultaneously to more than one conference or publication and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

How and Where to Submit

Authors are requested to submit full papers by Friday, February 11, 2000. Submitted papers should be no longer than 6,000 words (approximately 10 single spaced 8.5" x 11" pages), including figures, tables, and references. Excessively long papers will not be accepted.

Submissions will be accepted in electronic form only, in Word, PDF, or HTML format. The papers should be submitted via the Web submission form located on the conference Web site after January 10, 2000.

All submissions will be acknowledged by email within one week. If an acknowledgment is not received please send email to *usenix-win2000questions@usenix.org.*

Best Paper Awards

Awards will be given for the best paper and best student paper at the symposium.

Tutorials

On July 30 and 31, there will be tutorials in all areas and levels of expertise for Windows NT/2000 system engineers. Previous tutorial sessions have covered topics such as "Windows NT Security," "Windows NT Internals," "Configuring Samba, Avoiding Common Pitfalls," and "Administering Windows NT DHCP and DNS servers."

If you are interested in presenting a tutorial, please contact the USENIX tutorial coordinator:

Daniel V. Klein
Email : *dvk@usenix.org*

Demo and Poster Session

The symposium will include a session where, in an informal setting, participants can present and demonstrate their work. Information on submitting Demo & Poster session proposals will be made available in February on the symposium Web site:
http://www.usenix.org/events/usenix-win2000/.

Usage Abstracts

All symposium participants without an accepted paper will be requested to submit a one-page abstract or summary via the symposium web site at the time they register describing what they are doing or considering doing with Windows. These abstracts will be made available on the symposium web site before the symposium and will also be distributed in paper form to attendees. The abstracts are intended to facilitate communication among attendees, helping people find others with similar interests or problems.

Registration Materials

Materials containing all details of the symposium program, registration fees and forms, and hotel information will be available by May 2000. If you wish to receive the registration materials in print, please contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: *conference@usenix.org*

Questions

Please consult the symposium Web site (*http://www.usenix.org/events/usenix-win2000/*) for the latest information on this conference. If you have questions that are not addressed by the Web site, send email to *usenix-win2000questions@usenix.org.*

Announcement and Call for Papers **USENIX**

9th USENIX Security Symposium

Sponsored by USENIX in cooperation with The CERT Coordination Center

http://www.usenix.org/events/sec2000

August 14-17, 2000 **Denver, Colorado, USA**

Important Dates for Refereed Papers

Paper submissions due: *February 10, 2000*
Author notification: *March 23, 2000*
Camera-ready final papers due: *June 15, 2000*

Symposium Organizers

Program Co-Chairs
Steven Bellovin, *AT&T Labs—Research*
Greg Rose, *QUALCOMM Australia*
Program Committee
Carl Ellison, *Intel Corporation*
Ian Goldberg, *UC Berkeley*
Peter Gutmann, *University of Auckland*
Trent Jaeger, *IBM T.J. Watson Research Center*
Markus Kuhn, *University of Cambridge*
Marcus Leech, *Nortel*
Alain Mayer, *Lucent Technologies, Bell Laboratories*
Avi Rubin, *AT&T Labs—Research*
Jeff Schiller, *MIT*
Jonathan Trostle, *Cisco*
Wietse Venema, *IBM T.J. Watson Research Center*
Dan Wallach, *Rice University*
Tara Whalen, *Communications Research Centre Canada*
Elizabeth Zwicky
Invited Talks Coordinator
Win Treese, *Open Market Inc.*

Symposium Overview

The USENIX Security Symposium brings together researchers, practitioners, system administrators, system programmers, and others interested in the latest advances in security and applications of cryptography.

Dr. Blaine Burnham, director of the Georgia Tech Information Security Center (GTISC), will give the keynote address. Dr. Burnham most recently served as program manager for the National Security Agency (NSA) at Ft. Meade, Maryland.

If you are working in any practical aspects of security or applications of cryptography, the program committee would like to urge you to submit a paper. Submissions are due on February 10, 2000.

This symposium will last four days. Two days of tutorials will be followed by two days of technical sessions including refereed papers, invited talks, works-in-progress, panel discussions, and a two-day exhibition.

Symposium Topics

Refereed paper submissions are being solicited in all areas relating to system and network security, including but not limited to:

- Adaptive security and system management
- Analysis of malicious code
- Applications of cryptographic techniques
- Attacks against networks and machines
- Authentication and authorization of users, systems, and applications
- File and filesystem security
- Firewall technologies
- Intrusion detection
- IPSec and IPv6 security
- Public key infrastructure
- Rights management and copyright protection
- Security in heterogeneous environments
- Security incident investigation and response
- Security of agents and mobile code
- Techniques for developing secure systems
- Trust management
- World Wide Web security

Papers covering "holistic security"—systems security, the security of entire large application systems, spread across many subsystems and computers, and involving people and environment—are particularly relevant. On the other hand, papers regarding new cryptographic algorithms or protocols, or electronic commerce primitives, are encouraged to seek alternative conferences.

Refereed Papers

Papers that have been formally reviewed and accepted will be presented during the symposium and published in the symposium proceedings. The proceedings are provided free to technical session attendees. Additional copies will be available for purchase from USENIX.

Best Paper Awards

Awards will be given at the conference for the best paper and for the best paper that is primarily the work of a student.

Tutorials, Invited Talks, WIPs, and BoFs

In addition to the refereed papers and the keynote presentation, the technical program will include tutorials, invited talks, panel discussions, a Work-in-Progress session (WIPs), and Birds of a Feather Sessions. You are invited to make suggestions regarding

topics or speakers for any of these formats to the program chair via email to *securitychairs@usenix.org*.

Tutorials

Tutorials for both technical staff and managers will provide immediately useful, practical information on topics such as local and network security precautions, what cryptography can and cannot do, security mechanisms and policies, firewalls and monitoring systems.

If you are interested in proposing a tutorial, or suggesting a topic, contact the USENIX Tutorial Coordinator, Dan Klein, by phone at +1.412.422.0285 or by email to *dvk@usenix.org*.

Submitting an Invited Talk Proposal

These survey-style talks given by experts range over many interesting and timely topics. The Invited Talk Coordinator, Win Treese, welcomes suggestions for topics and requests proposals for particular talks. In your proposal state the main focus, including a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit via email to *securityit@usenix.org*.

Work-in-Progress Session (WIPs)

The last session of the symposium will be a Work-in-Progress session. This session will consist of short presentations about work-in-progress, new results, or timely topics. Speakers should submit a one- or two-paragraph abstract to *securitywips@usenix.org* by 6:00 pm on Wednesday, August 16, 2000. Please include your name, affiliation, and the title of your talk. The accepted abstracts will appear on the conference Web page after the symposium. The time available will be distributed among the presenters with a minimum of 5 minutes and a maximum of 10 minutes. The time limit will be strictly enforced. A schedule of presentations will be posted at the symposium by noon on August 17. Experience has shown that most submissions are usually accepted.

Birds-of-a-Feather Sessions (BoFs)

There will be Birds-of-a-Feather sessions (BoFs) both Tuesday and Wednesday evenings. Birds-of-a-Feather sessions are informal gatherings of persons interested in a particular topic. BoFs often feature a presentation or a demonstration followed by discussion, announcements, and the sharing of strategies.

How and Where to Submit Refereed Papers

Papers should represent novel scientific contributions in computer security with direct relevance to the engineering of secure systems and networks.

Authors must submit a mature paper in PostScript format. Any incomplete sections (there shouldn't be many) should be outlined in enough detail to make it clear that they could be finished easily. Full papers are encouraged, and should be about 8 to 15 typeset pages. Submissions must be received by February 10, 2000.

Along with your paper, please submit a separate email message in ASCII containing:

- The title, all authors of the manuscript, and their affiliations.
- The name of one author who will serve as a contact, with regular and electronic mail addresses, daytime and evening telephone numbers, and a fax number.
- Indicate any authors who are full-time students.

For more details on the submission process, authors are encouraged to consult the detailed author guidelines on the symposium website at: *http://www.usenix.org/events/sec2000/*.

All submissions will be judged on originality, relevance, and correctness. Each accepted submission may be assigned a member of the program committee to act as its shepherd through the preparation of the final paper. The assigned member will act as a conduit for feedback from the committee to the authors. Camera-ready final papers are due on June 15, 2000.

Authors will be notified of acceptance by March 23, 2000.

The Security Symposium, like most conferences and journals, requires that papers not be submitted simultaneously to another conference or publication and that submitted papers not be previously or subsequently published elsewhere. Papers accompanied by non-disclosure agreement forms are not acceptable and will be returned to the author(s) unread. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Specific questions about submissions may be sent to the program chairs via email to *securitychairs@usenix.org*.

For reliability, please send one copy of your paper to the program committee via **both** of the following two methods. All submissions will be acknowledged.

1. **Email (PostScript) to:**
   securitypapers@usenix.org
2. **Send a hard copy to:**
   Security Symposium
   USENIX Association
   2560 Ninth Street, Suite 215
   Berkeley CA 94710
   U.S.A.
   Phone: +1.510.528.8649

Security 2000 Exhibition

Demonstrate your security products to our technically astute attendees responsible for security at their sites. Meet with attendees in this informal setting and demonstrate in detail your security solutions. We invite you to take part. Contact: Dana Geffner, Email: *dana@usenix.org*, Phone: +1.831.457.8649

Registration Materials

Materials containing all details of the technical and tutorial programs, registration fees and forms, and hotel information will be available in May 2000. If you wish to receive the registration materials, please visit the symposium Web site or contact:

USENIX Conference Office
22672 Lambert Street, Suite 613
Lake Forest, CA 92630, USA
Phone: +1.949.588.8649
Fax: +1.949.588.9706
Email: *conference@usenix.org*

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITE

http://www.usenix.org

EMAIL

login@usenix.org

COMMENTS? SUGGESTIONS?

send email to jel@usenix.org

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *;login:* are December 1, 1999, and February 2, 2000.

USENIX The Advanced Computing Systems Association

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710