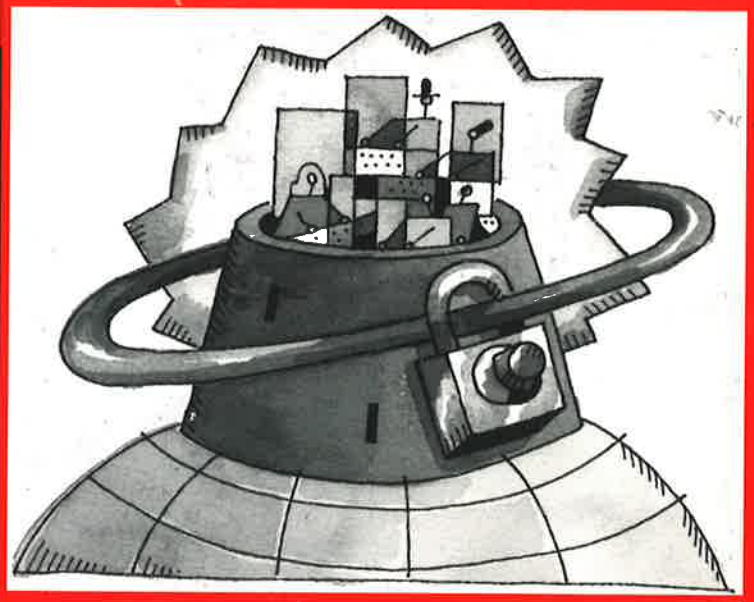
;login:

THE MAGAZINE OF USENIX & SAGE

December 2002 • volume 27 • number 6

Focus Issue: Security

Guest Editor: Rik Farrow



inside:

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

USENIX Upcoming Events

5TH NORDU/USENIX CONFERENCE (NORDU2003)

Sponsored by EurOpen.SE and USENIX

FEBRUARY 10–14, 2003
VÄSTERÅS, SWEDEN
http://www.nordu.org/NordU2003/

4TH USENIX SYMPOSIUM ON INTERNET TECHNOLOGIES AND SYSTEMS (USITS '03)

MARCH 26–28, 2003
SEATTLE, WASHINGTON, USA
http://www.usenix.org/events/usits03
Camera-ready final papers due: January 17, 2003

2ND CONFERENCE ON FILE AND STORAGE TECHNOLOGIES (FAST '03)

MARCH 1–APRIL 2, 2003
SAN FRANCISCO, CALIFORNIA, USA
http://www.usenix.org/events/fast03
Camera-ready final papers due: January 6, 2003

THE FIRST INTERNATIONAL CONFERENCE ON MOBILE SYSTEMS, APPLICATIONS, AND SERVICES (MOBISYS '03)

Jointly sponsored by ACM SIGMOBILE and USENIX in cooperation with ACM SIGOPS

MAY 5–8, 2003
SAN FRANCISCO, CALIFORNIA, USA
http://www.usenix.org/events/mobisys03/
Notification of acceptance: December 18, 2002
Camera-ready final papers due: March 4, 2003

THE 9TH WORKSHOP ON HOT TOPICS IN OPERATING SYSTMES (HOTOS IX)

Sponsored by USENIX in cooperation with IEEE TCOS

MAY 18–21, 2003
LIHUE, KAUAI, HAWAII, USA
http://www.usenix.org/events/hotos03/
Paper submission deadline: January 10, 2003
Notification of acceptance: March 17, 2003

2003 USENIX ANNUAL TECHNICAL CONFERENCE

JUNE 9–14, 2003
SAN ANTONIO, TEXAS, USA
http://www.usenix.org/events/usenix03/
Notification of Acceptance: January 22, 2003
Camera-ready final papers due: April 8, 2003

12TH USENIX SECURITY SYMPOSIUM

AUGUST 4–8, 2003
WASHINGTON, DC, USA
http://www.usenix.org/events/sec03/
Paper submissions due: January 27, 2003
Notification of Acceptance: March 20, 2003
Camera-ready final papers due: May 13, 2003

BSDCON '03

SEPTEMBER 8–12, 2003
SAN MATEO, CALIFORNIA, USA
http://www.usenix.org/events/bsdcon03/
Paper abstracts due: April 1, 2003
Notification of Acceptance: May 12, 2003
Camera-ready final papers due: July 8, 2003

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

contents

;login: Vol. 27 #6, December 2002

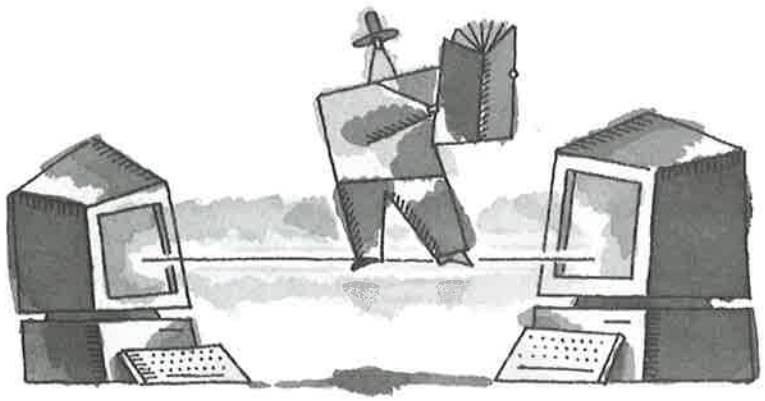
;login: is the official magazine of the USENIX Association and SAGE.

;login: (ISSN 1044-6397) is published bi-monthly 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 $60 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.



Art by Robin Jareaux

in this issue

by Rik Farrow

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

rik@spirit.com

I wanted to use an old photo to go with my comments for this, the fifth Security Special Edition of *;login:*. Why? This ID photo was taken in 1982, when I worked as a consultant for Morrow, an early designer and manufacturer of desktop computers. Morrow had brought in a security firm and had everyone get photo IDs, because they were losing a lot of money on stolen hard drives. In those days, a 34-megabyte hard drive cost nearly $2000 (when purchased in bulk) and had a high street value. Employees had been caught recovering hard drives from dumpsters – having previously put them into trash cans inside the building.

The addition of physical security made my life more difficult. I worked on two Morrow product lines, one that used CP/M and the other a version of UNIX. I had two hard drives, each with one OS on it, and would carry just the hard drives from my home office into Morrow. Getting administrative permission to do this was possible, but difficult.

My solution? Carry the hard drive in my backpack, under a spiral notebook. The guard would check my photo ID, look inside my backpack, which I would open wide for him. The heavy hard drive would sink beneath the notebook every time. So, to me, this old photo helps to remind me of the failings of security, even when it is as simple as a physical search.

At about the same time I was learning about UNIX by trying to teach other people about it; Tom Perrine was working with a group of people writing KSOS, a secure kernel designed to run under Version 6 UNIX. In his article, Perrine talks about how often today's programmers ignore lessons from the past. I did mention to Tom that the information that he wishes were better known used to be very secret. Many of us had copies of the Lion's notes (mine was at least 10th generation), which provided us with our *only* example of *any* operating system code or design. Things have gotten considerably better on this score, even if programmers are still working at re-inventing security.

Kevin Fu and his co-authors make the point that not everything about security was developed in the 1970s. Although they do leverage asymmetric encryption, a seventies invention, SFS provides true security but with a policy the opposite of most security policies: the remote users have control over their accounts, and anyone can decide to mount a public SFS share. SFS also incorporates other applications, including secure remote execution as an alternative to SSH and a read-only file system that uses digital signatures (making mirror sites much safer to use).

Jared Allison and George Jones, both from UUNET, provide practical discussions about taking advantage of logging and securing your network infrastructure, respectively. Two San Diego Supercomputing Center denizens wrote articles, one discussing computer legal practice (Erin Kenneally) and the other, how to start your own regional information watch (Abe Singer). Lance Spitzner, of the Honeynet Project, introduces HOSUS, a plan for getting early warnings about Internet attacks. And Sven Dietrich, of CERT, gets much more controversial with some of his suggestions for improving the security of Internet connected systems. And when it comes to the millions of poorly maintained Windows boxes attached to always-on connections these days, perhaps he has the right idea.

Ofir Arkin, of Sys-Security, presents us with just how insecure Voice over IP really is. Arkin has already come up with several advisories about vulnerabilities in different Internet phone products. Finally, Peter Salus describes his experiences during the Security Symposium.

This edition also includes the session summaries from the Security Symposium, something that I sincerely want to thank the summarizers for. One session did not get covered. A closing keynote had been scheduled, then a speaker substitution occurred, and then even the substitute flaked. Fortunately for USENIX, Dick Cheney was in town (San Francisco) to speak to some rich executives at a meeting in the Saint Francis Hotel. Not that Mr. Cheney wanted to speak to us, but his stand-in was more than willing.

The stand-in, a member of the San Francisco Mime Troupe (*http://www.sfmt.org*) lacked only a bevy of serious-looking Secret Service agents to complete the illusion. Ed Holmes gave a speech that, with the exception of several howlers, seemed just as authentic as any of the normal Administration double-speak. There were several people lined up to ask questions, starting with Peter Honeyman, who, for once, sounded to me completely inarticulate. John Gilmore, who has made a point of not traveling with photo ID, asked why he had to present ID to travel. The Cheney look-alike replied, "I don't have a problem traveling, what's wrong with you?" Holmes finished up by inducting the attendees into the Web Rangers, and enjoined them to report any "suspicious activities" they or their associates might be engaged in to the "government."

Some of what you will find in this issue comes from the hall talk – what goes on between sessions at any USENIX conference. But nothing can substitute for the experience and fun of actually being there.

One of the issues that I touched upon above, the millions of Internet-connected Windows systems, is likely to become an even more serious security problem over time. Today, the spread of worms and email viruses gets a lot of help from home systems. According to David Moore of CAIDA (at SDSC), Code Red II is still around, still running on systems that appear to be primarily running at home, in small offices, and in Southeast Asia (based on the times the scans peak). Based on a recent virus report, Nimda is still infecting systems (4.7% of infections), with SirCam at 10.4% of virus occurrences. Both of these involve bugs in Internet Explorer that should have been patched years ago.

The issue with these persistent pests is not so much their nuisance value, but how to go about patching systems that belong to people who either will not or cannot patch them. In many cases, these systems have pirated copies of MS software installed, and their owners may be afraid to visit Microsoft and download the patches. Or the owners remain blissfully unaware that a lot of their network bandwidth goes to scanning and spamming other sites around their world.

It's not just Windows users, either. Installations of Linux from old CDs remain one of the biggest targets of automatic scanning today (my firewall logs show that DNS probes are second only to probes looking for open Windows file shares).

The problem of patching systems goes beyond the systems under our control. Sven Dietrich goes so far as advocating attacking "the immediate surroundings of the attacker," or using a version of an attack, like Code Red, to install the correct patch (Code Green). While such attacks violate the laws of most countries, we might someday reach the point where they will seem justified. After receiving 176 spam emails in a single night (92% spam that night) and seeing my firewall logs filled with denied probes, I am beginning to believe that Sven has a good point.

motd

by Rob Kolstad

Rob Kolstad is currently Executive Director of SAGE, the System Administrators Guild. Rob has edited *;login:* for over ten years.

kolstad@sage.org

I Really Hate to Complain, But . . .

I am so tired of spam. It was driving me to distraction before my local sysadmin installed a spam-mitigation system (doesn't matter which one – it seems to do an OK job – and I'm not here to start a religious fight).

It just hasn't been that long since every incoming message rang a bell, and the Pavlovian feeling that bell evoked was like Christmas: another exciting gift to be unwrapped to be enjoyed. I know I'm sounding mawkishly AOL-ian here, but I really do like hearing from people and getting good e-mail.

As time wore on, the bells only signaled new irritation. I even turned off the bell for a while, but my life just felt so empty.

Enough pundits have already justified their spending habits, money-making habits, and adequate size of their body parts that I don't need to elaborate here on what a waste of time, energy, and bandwidth spam is. I just can't figure out who is motivating the spammers to continue their merry little game: stupid entrepreneurs? stupid consumers? I just don't get it.

But the biggest thing I don't get is this: Why is it that so little is being done by the technological community to discourage spamming in the first place? Oh, there's the occasional legislation here or there, but I really have in my mind that spammers should be treated as societal outcasts and shunned as "bad citizens."

Over and over I hear the trite responses, "Oh, we need to level the playing field with the big (read: well-funded) corporations for advertising," or my local newspaper's favorite: "Just delete the spam; how much time could it take?" ARRGH! It takes a lot of time! Important messages are buried. Messages demanding quick response are lost amid the noise.

Why aren't sysadmins circulating petitions against spam? Why aren't users screaming to get the problem solved? Why are some sites still accepting all mail on the off-chance some mis-classified spam-mail will actually be a sales-lead? Hotmail reports rejecting one BILLION spam messages per day. Of the non-rejected (and delivered) messages, 80% are spam (undetected earlier, sad to say).

Why do people put up with this?

I'd like to hear from you why you are accepting the situation, if you are. I have an idea that we're collectively spending thousands (hundreds of thousands?) of person-hours fighting this scourge (this or that spam filter, configuring mail systems, etc.) – but why? Why is it allowed to happen in the first place? Do people with small body-parts wait at their keyboards hoping someone will mail them with a solution? Are people really making money too slowly? Have I really missed the boat for transfer of the ill-gotten Oil Minister's gains from Nigeria? I think not. So why does it continue? Help!

using SFS for a secure network file system

by Kevin Fu,

Kevin Fu is a doctoral student at the Laboratory for Computer Science at MIT. His research interests are computer security, cryptography, and operating systems.

fubob@mit.edu

Michael Kaminsky,

Michael Kaminsky is a doctoral student at the Laboratory for Computer Science at MIT. His research interests are operating systems, security, and networking.

kaminsky@lcs.mit.edu

and David Mazières

David Mazières is an assistant professor of computer science at New York University. He began the SFS project while a doctoral student at MIT.

dm@cs.nyu.edu

Introduction

The Self-Certifying File System (SFS) (*http://www.fs.net/*) is a secure distributed file system, and associated utilities, that can both increase the security of networks and simplify system administration. SFS lets people securely access file systems over insecure networks without the need for virtual private networks (VPNs) or a public key infrastructure (PKI). In many situations, SFS provides a suitable and more secure alternative to the widely deployed NFS file system. Nonetheless, SFS gracefully coexists with NFS and other file systems such as Samba. Thus, one can easily install, test, and incrementally deploy SFS without disrupting existing network services.

SFS administration is greatly simplified by the fact that client machines have no configuration options. An SFS client just needs to run a program called sfscd at boot time; users can then access any server with no further administrative assistance. In contrast, many distributed file systems require the client to have a list of what remote file systems to mount where. In SFS, it is actually the server that determines which of its file systems clients mount on what pathnames. As users access those pathnames, client machines learn of new servers and transparently "automount" them.

SFS cryptographically secures all client-server network communications with encryption and a message authentication code. To prevent rogue servers from impersonating valid ones, each SFS server has a public key. A server's files all reside under a so-called *self-certifying pathname* derived from its public key. Self-certifying pathnames contain enough information for an SFS client to connect to a server and establish a cryptographically secure channel – even if the client has only just learned of the server through a user's file access. A variety of techniques exist for users to obtain servers' self-certifying pathnames securely.

In addition to protecting network traffic, SFS performs user authentication to map remote users to credentials that make sense for the local file system. SFS's user-authentication mechanism is based on public key cryptography. On the client side, an unprivileged *agent* process holds a user's private keys and transparently authenticates him or her to remote servers as needed. On the server side, SFS keeps a database of users' public keys. Users can safely register the same public key in multiple administrative realms, simplifying the task of accessing several realms. Conversely, administrative realms can also safely export their public key databases to each other. A file server can import and even merge user accounts from several different administrative realms.

Background

The SFS project began in 1997 after one of the authors became frustrated by the lack of a global, secure, decentralized network file system. No existing file systems had all three properties.

NFS does not have a viable notion of security for most environments. NFS essentially trusts client machines, allowing an attacker to impersonate a legitimate user with little effort. NFS also lacks a global namespace, because client administrators can mount

NFS file systems anywhere. The same files might be accessible under two different paths on two different machines.

The Andrew File System (AFS) offers a global namespace and, in some contexts, security, but it cannot guarantee data integrity to users who do not have accounts in a server's administrative realm. Moreover, when users have several accounts in different realms, AFS makes it hard to access more than one account at a time. As a result, AFS tends to lead to inconveniently large administrative realms – often as large as an entire campus. Such unwieldy realms in turn restrict users who might want greater autonomy. For example, it is not uncommon for AFS users to have to recycle "guest" accounts to avoid the onerous burden of going through a central administration to create a new account for every visitor. SFS, in contrast, lets a server operator both import a campus-wide user database and manage additional local accounts.

NFS does not have a viable notion of security for most environments.

STATUS

SFS is free software and runs on UNIX operating systems that have solid NFSv3 support. We have run SFS successfully on recent versions of OpenBSD, FreeBSD, MacOS X, OSF/1, Solaris, and several Linux distributions. Although a Windows port exists, it relies on a commercial NFS implementation. We currently have no plans to merge the Windows port into the mainline distribution, though we would like to see this happen eventually.

There are pre-packaged SFS distributions available for Debian, Red Hat, and FreeBSD, among others. The packages greatly simplify installation, because SFS requires a robust compiler that can cope with extensive use of C++ templates. (Some versions of GCC generate internal compiler errors when compiling SFS from scratch.) Because SFS is open source, US regulations allow export of the code to most countries.

The SFS installation and setup procedures are described below. The SFS Web site (*http://www.fs.net/*) has technical papers which provide a detailed discussion of related work, a performance analysis, and the theory behind SFS.

OVERVIEW

From a system administrative point of view, SFS consists of two programs run at boot time. SFS clients must run the SFS client daemon (`sfscd`), which creates the `/sfs` directory and implements the auto-mounting of remote SFS servers. SFS servers must run the SFS server daemon (`sfssd`), which makes local file systems available to SFS clients on TCP port 4.

Internally, however, SFS is structured in a modular manner and consists of many daemons. `sfscd` and `sfssd` each launch and communicate with a number of subsidiary daemons that actually implement the different SFS features (Figure 1). For example, `sfscd` is responsible for automatically mounting new remote file systems. On the server machine, `sfssd` accepts incoming SFS connections and de-multiplexes these requests to the appropriate SFS server daemons. The three basic servers are the SFS file server, the authentication server, and the remote login server. The client and server file system daemons communicate with the kernel using NFS loopback. SFS components communicate with each other using RPC. SFS implements a secure RPC transport which provides confidentiality and integrity. Finally, the SFS agent runs on the client machine, one instance per user.

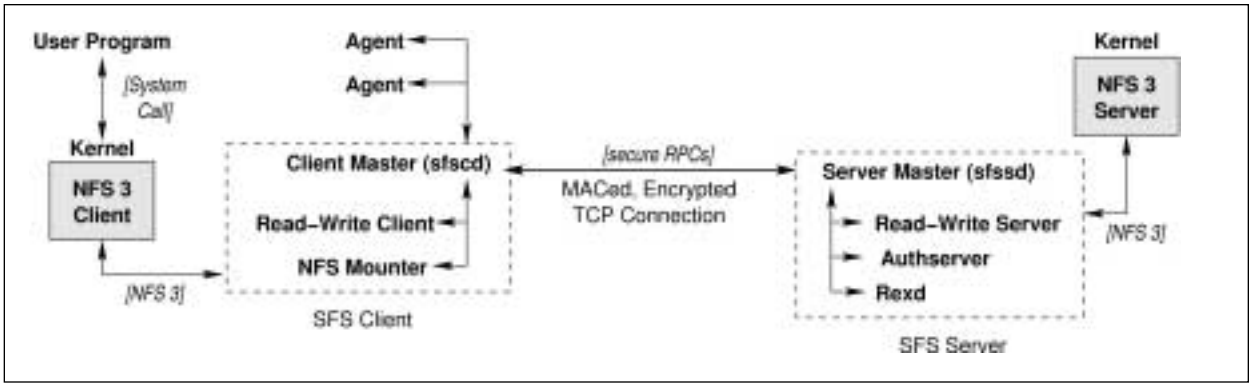


Figure 1: Architectural overview of SFS

If you already use NFS, switching to SFS is straightforward. Because the SFS server can coexist with an NFS server on the same machine, you don't have to switch "cold turkey" from NFS to SFS: you can serve both from the same machine. For instance, you may wish to continue using NFS for local machines but make SFS available for users traveling or using wireless networks.

A single machine can be both an SFS client and a server. However, the client software will refuse to mount an SFS file server running on the same machine, since this can cause deadlock in the kernel on some operating systems. This is a result of SFS's implementation as an NFS loopback server for portability.

Naming Servers

A fundamental problem that SFS tries to address is how to name resources securely. Setting up secure communication requires authentication of the remote resource; systems today often use some form of public key cryptography. The basic problem facing these public key–based systems is key distribution: how does the user who wants to connect securely to a remote resource get that resource's public key securely?

Self-Certifying Pathnames

SFS does not mandate any particular kind of key distribution, but instead provides a flexible set of options based on self-certifying pathnames. Given a self-certifying pathname, the public key of the file server can be certified with no additional information. Thus, if one obtains a self-certifying pathname in a trusted manner, the SFS client will automatically verify the associated server's public key.

Figure 2 shows the two basic components of self-certifying pathnames: the location and HostID. The location is the DNS hostname of the file server; it tells the client software where to find the server, but not how to communicate securely with it. The

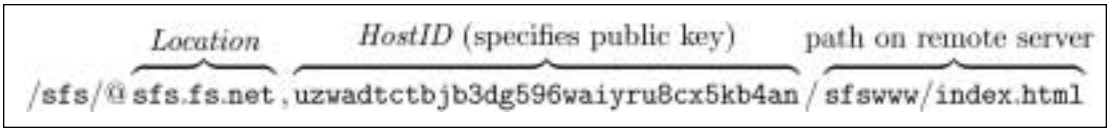


Figure 2: A self-certifying pathname

HostID portion of the self-certifying pathname is what allows secure communication. The HostID is a SHA-1 hash of the server's public key, similar in concept to a PGP fingerprint. Hence, the client can insecurely ask the server for its public key, then verify

that the public key actually matches the hash in the pathname. A user who trusts in the authenticity of the HostID can then trust in the authenticity of the corresponding public key that the client has fetched and verified.

When a user accesses a self-certifying pathname (under /sfs), the SFS client software contacts the server based on the location component. The server responds with its public key. If the server can prove its identity by demonstrating it has the corresponding private key, the SFS client will automatically mount the file server under /sfs. All communication between the client and server is encrypted and mutually authenticated. We will discuss user authentication shortly.

While self-certifying pathnames require some method for users to obtain HostIDs securely, they are not wedded to any particular public key infrastructure or method of key distribution. The user is free to decide, even on a host-by-host basis, how to obtain the HostID of a server. Users who need a centralized repository of public keys (a certification authority) and those who want to set up stand-alone servers both benefit from self-certifying pathnames. The following sections demonstrate several useful techniques for securely obtaining servers' self-certifying pathnames.

Symbolic links provide an easy way for users to refer to a self-certifying pathname.

Symbolic Links as Simple Certificates

Symbolic links provide an easy way for users to refer to a self-certifying pathname. For example, a user might create a symbolic link as follows:

```
redlab -> /sfs/@redlab.lcs.mit.edu,gnze6vwxtwssr8mc5ibae7mtufhphzsk
```

When the user accesses the path redlab, the SFS client software will mount the file server at *redlab.lcs.mit.edu* with the given public key hash. Symbolic links allow users to assign human-readable names to hard-to-type self-certifying pathnames. The user could store these links in a subdirectory of his or her home directory, and they would serve a similar function to the user's SSH known_hosts file.

Administrators can use symbolic links to provide a system-wide set of names for commonly used file servers. These may be distributed using a floppy disk, rsync/rdist over SSH, or whatever other technique is in use for remotely administering software on machines. Once a single symbolic link has been installed on a client, administrators can bootstrap a larger set of self-certifying pathnames and symbolic links from a trusted source. These links can be placed in a well-known location (e.g., /sfslinks) and would serve a similar function to a host-wide SSH /etc/ssh/ssh_known_hosts file.

Symbolic links in the file system have an advantage over a known_hosts-style name cache, because a name lookup can involve several levels of symbolic links; moreover, these symbolic links can reside on multiple file systems, some of which are on SFS. This feature of symbolic links and self-certifying pathnames provides a convenient, simple way to implement certificate authorities as file servers. As a hypothetical example,

```
/sfs-CAs:
        sun -> /sfs/@sfs.sun.com,ytzh5beann4tiy5aeic8xvjce638k2yd
        thawte -> /sfs/@thawte.com,...
        ...

/sfs/@sfs.sun.com,ytzh5beann4tiy5aeic8xvjce638k2yd:
        yahoo -> /sfs/@www.yahoo.com,...
        redhat -> /sfs/@redhat.com,...
        ...
```

Certification programs provide the user with a lot of flexibility.

If a user wants to read from the file server which certificate authority Sun calls redhat, the user can reference a path such as:

```
cat /sfs-CAs/sun/redhat/README
```

The directory /sfs-CAs is on the local machine, the directory /sfs-CAs/sun is on the machine sfs.sun.com, and the directory /sfs-CAs/sun/redhat is on the machine redhat.com (as is the file README). The pathname by which one accesses a file determines how the file server is authenticated. For example, the same README file might be available as:

```
/sfs-CAs/thawte/redhat-server/README
```

Server Nicknames

Some users might require a more sophisticated means of naming a server. SFS allows users to invoke arbitrary certification programs to map a human-readable "nickname" into a self-certifying pathname. The user's agent keeps a list of these certification programs and runs them as needed.

Server nicknames are non-self-certifying names that a user accesses under /sfs (e.g., /sfs/work or /sfs/lab-machine1). When the SFS client software sees a nickname, it asks the user's agent to translate the nickname into a self-certifying pathname. The agent will invoke, in order, all of the certification programs that the user has registered with it until there is a successful match. The sfskey utility, discussed in the examples, allows users to register certification programs with their agents.

Certification programs provide the user with a lot of flexibility. For example, the certification program might look up the nickname in an LDAP database. As a less complex example, the SFS distribution actually comes with a program called dirsearch that takes a list of directories and looks up the nickname in each one until it finds a symbolic link with that name. For instance, the certification program

```
dirsearch ~/my-links /sfslinks/sfs.mit.edu /sfs-CAs/sun
```

would have the effect of giving precedence to the user's personal links directory first, then some university-wide directory, and finally Sun's. The dirsearch program allows the user to specify his or her own trust policy.

Secure Remote Password Protocol (SRP)

SFS also provides a means of securely downloading a server's self-certifying pathname with a password. In this case, the password typed by the user is actually used to authenticate the server to the user in addition to the more conventional authentication of user to server. Though users cannot be expected to remember self-certifying pathnames, they will remember passwords of their own choosing. In this way, users can always resort to typing their passwords if there is not a more convenient way of obtaining a server's pathname.

SFS uses the Secure Remote Password Protocol (SRP) for password authentication.[1] SRP allows users both to download self-certifying pathnames securely and to download encrypted copies of their own private keys. When a user registers a public key with an SFS server, the user can additionally give the server an encrypted copy of the corresponding private key and a secret "SRP parameter" computed as a function of his or her password and the server's location. When the user and server later engage in the SRP protocol, SRP mutually authenticates the two sides' communications. The details

1. Thomas Wu, "The Secure Remote Password Protocol," *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, San Diego, CA, March 1998, pp. 97-111.

of SRP are handled for the user by the sfskey program (or indirectly through sfsagent, which can invoke sfskey). After a successful run of SRP, sfskey installs a symbolic link from the server's location to its self-certifying pathname. For example:

```
/sfs/redlab.lcs.mit.edu ->
/sfs/@redlab.lcs.mit.edu,gnze6vwxtwssr8mc5ibae7mtufhphzsk
```

We chose SRP rather than a simpler protocol so as to protect weak passwords against offline "dictionary attacks." In other file systems, such as AFS, an attacker can exchange a single message with the server, then invest a large amount of computation to guess and verify an unbounded number of candidate passwords. SRP allows only "online" password-guessing attacks – the number of passwords an attacker can guess is proportional to the number of messages the attacker has exchanged with the server or user.

Examples

Many operating system distributions offer pre-compiled SFS binary packages. These packages provide both client and server support out-of-the-box. Other users will most likely have to compile and install SFS from the source. The SFS Web site contains details about the compilation process and about compiler compatibility. Because the client software is implemented as an NFS loopback server, all SFS installations require working NFSv3 client support.

Installing the SFS Client

Installing the SFS client is straightforward using package management tools such as dpkg or RPM. If you are behind a firewall, you will need to allow outgoing connections to TCP port 4. The following example shows how to set up SFS on a freshly installed Red Hat 7.3 box:

```
[root@client /root]# rpm -ivh sfs-0.7-1.i386.rpm
Preparing...  ########################################### [100%]
   1:sfs      ########################################### [100%]
[root@client /root]# /etc/rc.d/init.d/sfscd start
Starting sfscd:                                    [  OK  ]
```

SFS clients require no configuration. Simply run the program sfscd as shown above, and a directory /sfs should appear on your system. To test your client, access our SFS test server. Here we download a file:

```
$ cd /sfs/@sfs.fs.net,uzwadtctbjb3dg596waiyru8cx5kb4an
$ cat CONGRATULATIONS
You have set up a working SFS client.
```

Note that the /sfs/@sfs.fs.net:... directory does not need to exist before you run the cd command. SFS transparently mounts new servers as you access them.

User Authentication

In the example above, the SFS server is exporting its file system publicly and read-only. Typically, however, SFS servers will require user authentication, so that only registered users can access the file system. A registered user is one whose public key has been installed on the SFS server: specifically, in the "authserver." To register a public key, log into the file server and run the command:

```
$ sfskey register
```

Many operating system distributions offer pre-compiled SFS binary packages.

SRP allows the user to securely download a copy of the server's self-certifying hostname.

This will create a public-private key pair for you and register the public key with the server. Note that if you already have a public key on another server, you can reuse that public key by giving sfskey your identifier at that server (i.e., sfskey register user@other.server.com).

The SFS user registration process also sets up SRP. As mentioned above, SRP allows the user to securely download a copy of the server's self-certifying hostname. SFS also uses SRP to store an encrypted copy of the user's private key on the server. The user can then download a copy of his or her private key from the server knowing only a password. Because the private key is encrypted, the server does not have access to it.

In some settings, users do not have permission to log into file servers and thus cannot run the sfskey register command. In this case, there are two options for creating user accounts. The system administrator can ask the user to supply a public key, or else can ask the user for an initial password and use the password to register a temporary public key and SRP parameter.

Running the SFS User Agent

Once you have registered your public key with an SFS server, you must run the sfsagent program on an SFS client when you wish to access the server. On the client, run the command:

```
$ sfsagent user@my.server.com
Passphrase for user@my.server.com/1024:
```

my.server.com is the name of the server on which you registered. user is your identifier on that server. (The value 1024 is the size in bits of SRP's cryptographic parameters, which you can ignore, though paranoid users may wish to avoid small values.) This command does three things: it runs the sfsagent program, which persists in the background to authenticate you to file servers as needed; it fetches your private key from the server and decrypts it using your password and SRP; and, finally, it fetches the server's public key and creates a symbolic link from /sfs/my.server.com to /sfs/@my.server.com,HostID. Each user has a different view of the /sfs directory. Thus, one user's links in /sfs will not be visible to another user, and two users' links will not interfere with each other.

If, after your agent is already running, you wish to fetch a private key from another server or download another server's public key, you can run the command:

```
$ sfskey add user@myother.server.com
Passphrase for user@myother.server.com/1024:
```

In fact, sfsagent runs this exact command for you when you initially start it up. Note that sfskey does not take a self-certifying pathname as an argument; the user's password and SRP are sufficient to authenticate the server holding your encrypted private key.

To generate a public-private key pair explicitly and save it to disk, use the following command:

```
$ sfskey gen
```

Optional arguments allow you to specify the key size and name. The sfskey subcommands edit, list, delete, and deleteall manage the keys stored in your SFS agent. The

sfskey update command allows you to replace the key stored on a server with a new one.

When you are done using SFS, you should run the command

```
$ sfskey kill
```

before logging out. This will kill your sfsagent process running in the background and get rid of the private keys it was holding for you in memory. There is also an option to specify a timeout to automatically remove keys from memory.

Setting Up an SFS Server

Setting up an SFS server requires very little configuration. The procedure consists of setting up the NFS loopback and choosing what directory trees to export. For extra security, you may wish to configure local firewall rules to prevent non-local users from probing portmap. Recall that only the server itself needs to access the NFS exports. Here we start with a Red Hat 7.3 box that has the SFS client software already installed.

```
[root@server /root]# rpm -ivh sfs-servers-0.7-1.i386.rpm
Preparing...          ########################################### [100%]
      1:sfs-servers   ########################################### [100%]
[root@server /root]#
```

Let's assume you want to export disks mounted on /disk/disk0 and /disk/disk1 to authorized users. We first need to create mount points for the SFS root file system and for each exported directory tree.

```
[root@server /root]# mkdir -p /var/sfs/root/disk0 /var/sfs/root/disk1
```

Recall that for portability reasons, SFS accesses the file systems it exports by pretending to be an NFS client over the loopback interface. Thus, the server must export the desired file systems to the localhost via NFS. We add the following three lines to the /etc/exports file to enable NFS exporting of /var/sfs/root, /disk/disk0, and /disk/disk1 to the localhost:

```
[root@server /etc]# cat  /etc/exports
/var/sfs/root localhost(rw)
/disk/disk0 localhost(rw)
/disk/disk1 localhost(rw)
```

Now we start the NFS server:

```
[root@server /etc]# /etc/rc.d/init.d/portmap status
portmap (pid 643) is running...
[root@server /etc]# /etc/rc.d/init.d/nfs start
Starting NFS services:                     [  OK  ]
Starting NFS quotas:                       [  OK  ]
Starting NFS mountd:                       [  OK  ]
Starting NFS daemon:                       [  OK  ]
```

We're almost done! Now we need to create the server's public-private key pair and tell the SFS server to also export the same directories:

```
[root@server /root]# sfskey gen -P /etc/sfs/sfs_host_key
Creating new key for /etc/sfs/sfs_host_key.
      Key Name: root@my.server.com

sfskey needs secret bits with which to seed the random number generator.
Please type some random or unguessable text until you hear a beep:
```

Setting up an SFS server requires very little configuration.

```
DONE
[root@server /root]# cat  /etc/sfs/sfsrwsd_config
Export /var/sfs/root / R
Export /disk/disk0 /disk0
Export /disk/disk1 /disk1
```

The R flag makes /var/sfs/root globally readable – you can omit this flag if you do not wish to have any anonymously accessible directories. Finally, we start the server:

```
[root@server sfs]# /etc/rc.d/init.d/sfssd start
Starting sfssd:                                            [  OK  ]
[root@server sfs]# tail /var/log/messages
Sep 12 23:46:43 server sfssd: sfssd startup succeeded
Sep 12 23:46:43 server : sfssd: version 0.7, pid 1881
Sep 12 23:46:43 server : sfsauthd: version 0.7, pid 1885
Sep 12 23:46:44 server : sfsauthd: serving
@server.mit.edu,66zrwhw5i9jr5ym7i9mkcxijn5fmtaz8
Sep 12 23:46:44 server : sfssd: accepted connection from 18.247.7.168
Sep 12 23:46:44 server rpc.mountd: authenticated mount request
        from localhost.localdomain:790 for /disk/disk0 (/disk/disk0)
Sep 12 23:46:44 server rpc.mountd: authenticated mount request
        from localhost.localdomain:790 for /disk/disk1 (/disk/disk1)
Sep 12 23:46:44 server rpc.mountd: authenticated mount request
        from localhost.localdomain:790 for /var/sfs/root (/var/sfs/root)
Sep 12 23:46:44 server : sfsrwsd: version 0.7, pid 1886
Sep 12 23:46:44 server : sfsrwsd: serving /sfs/@server.mit.edu,
        66zrwhw5i9jr5ym7i9mkcxijn5fmtaz8
```

Remote Login

REX is an SSH-like remote login tool that uses self-certifying paths instead of a static known_hosts file. While REX can be disabled on the server by commenting out a single line in the sfssd_config configuration file, it makes remote login more pleasing to users with home directories stored in SFS. By default, REX forwards X11 connections and forwards the SFS agent itself. The basic invocation is with a hostname:

```
$ rex amsterdam.lcs.mit.edu
rex: Prepending '@' to destination 'amsterdam.lcs.mit.edu' and attempting SRP
Passphrase for fubob@amsterdam.lcs.mit.edu/1024:
rex: Connecting to @amsterdam.lcs.mit.edu,bkfce6jdbmdbzfbct36qgvmpfwzs8exu
amsterdam:(~/)%
```

REX can use the SFS agent and/or SRP to map DNS hostnames to self-certifying pathnames as described earlier. In the above example, REX prompts for a password and uses SRP to obtain amsterdam's self-certifying pathname and the user's private key securely. Subsequent logins to the same server do not require a password.

REX also accepts other names for servers. For example, REX accepts self-certifying pathnames and even SFS symbolic links as a way of naming servers. System administrators may find fully qualified self-certifying pathnames useful for non-interactive scripts. Here we generate a list of currently logged-in users:

```
$ rex @amsterdam.lcs.mit.edu,bkfce6jdbmdbzfbct36qgvmpfwzs8exu
/usr/bin/who
```

Connection caching allows subsequent REX executions to the same server to avoid public key operations. The client and server generate a new session key based on the

previous one. The speedy reconnection will be useful for system administrators who frequently make multiple remote execute commands to the same servers. No longer will each command require a good portion of a second to complete.

You can see what sessions your agent currently maintains by running

```
$ sfskey sesslist
```

The additional command sfskey sesskill removes a connection from the agent's cache.

SSH was the main inspiration for REX, as we needed an SSH-like tool that could work with SFS. Although we could have extended SSH for this purpose, SSH servers typically read files in users' home directories during user authentication. This policy is incompatible with our goal of integrating remote login with a secure file system, as the remote login machine would generally not have permission to read users' files before those users are authenticated.

For those who are hesitant to use REX but need remote login to work with home directories stored in SFS, the libpam-sfs module may be a reasonable alternative.

SFS Toolkit

The SFS file system infrastructure or "toolkit" provides support for several other projects and extensions. Below is a short list of what the extended SFS world has to offer. For all the details and references, see the SFS Web page.

File Systems

One of the goals of SFS is to facilitate the less painful development of new file systems. As an example, one of the authors wrote a crude encrypted file system in just 600 additional lines of C++ code. SFS provides an asynchronous library and an efficient NFS loopback server. By implementing the server side of the NFS loopback server, a developer can create a new file system that works on all operating systems with solid NFSv3 support.

In a read-only dialect of SFS (TOCS 20(1)), a publisher could replicate content on untrusted servers while maintaining the integrity of the file system as a whole. The read-only dialect is significantly faster than the read-write dialect, because the server performs no online cryptography. Another dialect caters to low-bandwidth connections (SOSP 2001). The Chord system (SIGCOMM 2001) uses the SFS asynchronous library to implement a peer-to-peer lookup algorithm. The Cooperative File System (SOSP 2001) uses the SFS asynchronous library to implement a distributed peer-to-peer read-only file system. Ivy (OSDI 2002) does the same for a read-write log-based file system.

ACLs

One drawback to traditional UNIX file sharing is the lack of access control lists (ACLs). We find that file sharing in NFS (and hence SFS) is limited because of the difficulty of creating and administering groups for users without administrative privileges. In a dialect of SFS under development, the server supports ACLs similar to those of AFS. A key difference is that we have no-hassle cross-realm support. A student at MIT can give a friend at NYU access to a file or directory simply by adding the friend's public key to the appropriate ACLs. Although the ACL prototype works and appears in the main sourcetree, it does not yet appear in an official release.

One of the goals of SFS is to facilitate the less painful development of new file systems.

. . . we have never lost a file to SFS.

Caveat

SFS serves files to clients using cryptographically protected communications. As such, SFS can help eliminate security holes associated with insecure network file systems and let users share files where they could not do so before. That said, we realize that perfect security is a remote fantasy. Our documentation on the SFS Web site discusses many of the security issues raised by SFS.

Conclusion

Our research groups have used SFS on a daily basis for several years. Several of us use SFS for our home directories. In part because SFS is implemented on top of mature NFS code, we have never lost a file to SFS. A number of groups outside of our research groups also use SFS in production environments. We hope you find SFS as convenient and useful as we have.

ACKNOWLEDGMENTS

Several people have contributed to the SFS development, including Chuck Blake, Benjie Chen, Frank Dabek, David Euresti, Kevin Fu, Frans Kaashoek, Michael Kaminsky, Maxwell Krohn, David Mazières, Eric Peterson, George Savvides, and Nickolai Zeldovich. We thank Eric Anderson, Sameer Ajmani, Michael Barrow, Rik Farrow, Maxwell Krohn, Chris Laas, Emil Sit, and Ram Swaminathan for helpful feedback on drafts of this article.

SFS is funded by the DARPA Composable High Assurance Trusted Systems program under contract #N66001-01-1-8927. Support also came from an NFS Graduate Fellowship and a USENIX Scholars Research Grant.

automated log processing

by Jared Allison

Jared Allison is an Internet security engineer at UUNet. He has spent much of his time working on projects that automate traffic analysis, intrusion detection, and router and system security configuration checking in large-scale networks. Most of his work is done with C, Perl, and PHP.

jallison@UU.NET

Hindsight Is 20/20

It happens to everyone – one of your systems is compromised and you go back through the logs to see what happened. How did the intruder break in? Then you see it:

```
Jun 14 13:08:58 computer.company.com sshd[9779]: log: Connection from
452.312.34.58 port 3552
Jun 14 13:09:01 computer.company.com sshd[9779]: fatal: Local: crc32
compensation attack: network attack detected
```

(Names and numbers have been changed to protect the innocent.)

As you trace it through the logs, you can see the whole sequence of events. But why didn't you notice this *before* it happened? Or at least quickly afterward? When going through log files to determine what happened and what needs to be fixed, it is often easy to see entries that indicated an attempted intrusion. If only you'd seen those logs beforehand! Of course, nobody has the time to constantly watch their logs for scans, dictionary attacks, misconfigured equipment, and so on. Even given the time, it would be very difficult to pick these events out of the noise. To have a chance at keeping up, monitoring tasks must be automated with scripts that can filter out the noise and report only information that is relevant and useful. Though there is some time delay inherent in log processing, logs can still be useful in proactive security. Let's look at some straightforward examples of how to use logs to help prevent intrusions and increase the visibility of events on your network.

Log Files and Low-Hanging Fruit

Simple port scans are one of the most common and easy-to-detect indicators of wrongdoing. This is especially true of worms and IRC "botnets" that don't attempt to be stealthy about their scanning. Using an intrusion detection system to watch for attempted break-ins can be very effective and has been treated in many books. When an IDS isn't an option, however, you can still detect a lot of the simple intrusion attempts through log analysis. Firewall logs from access lists, for example, can provide enough information to detect port scans (protocol, source and destination IPs and ports, and times). The question is, what to do with that information? How do you pick out the scans from the noise? Once you pick out the scans, then what do you do?

Before you can take any action, you have to find the scan activity. The first step toward doing that is to parse your logs into a usable format. Perl is great for this. You are going to need to be able to search through and group the data a number of different ways. Perl alone will work for this as well, but an easier and more flexible solution is to use a database. Writing your Perl parsing script to dump the interesting part of the logs in, say, CSV format and loading the whole file into the database at once is much faster than interfacing directly to the database and loading one row at a time. These scripts are very simple and generally look something like this:

```
while( defined($line=<LOGFILE>) ) {
    if( $line =~    /(\d+)-(\d+)\s+                  # date
                     (\d+:\d+:\d+)\s+:                # time
                     (\S+)\s+                         # protocol
                     (\d+\.\d+\.\d+\.\d+):(\d+)\s+     # source IP and port
                     ->\s+
                     (\d+\.\d+\.\d+\.\d+):(\d+)       # destination IP and port
                    /x ) {
```

```
        print TEMPFILE ( "$1-$2 $3", (getprotobyname($4))[2], ",",
                unpack( "N", inet_aton($5) ), ",$6,",
                unpack( "N", inet_aton($7) ) ",$8, "\n" );
        }
}
```

The particular regular expression will vary depending on the format of the log files. Finally, you should define a generalized table layout so that you can import logs from many different types of devices (e.g., routers, firewalls, or even tcpdump) into the same database table. At a minimum, your database table should have the following fields:

Data Source	(This field could be text – perhaps an enumerated type.)
Date & Time	(Store these in the same field if possible.)
Protocol	(Use /etc/protocols to convert this to an 8-bit unsigned integer.)
Source IP Address	(Store as a 32-bit unsigned integer.)
Source Port	(Store as a 16-bit unsigned integer.)
Destination IP Address	(Store as a 32-bit unsigned integer.)
Destination Port	(Store as a 16-bit unsigned integer.)
Action	(Was this packet permitted or denied?)

A quick note on ICMP – it is useful to store the ICMP type and code information. An easy way to do this without wasting space is to store them in the source port and destination port fields, respectively. Just remember to check the protocol in your scripts, or you might spend a lot of time trying to track down why anyone would use those weird low ports.

Once your data is loaded into a database, a fairly easy way to look for obvious scans is to group the data as follows. Grouping by the source and destination IP address and then looking for large numbers of different destination ports will reveal potential host scans, since a conversation between two hosts is usually confined to, at worst, a small number of ports. Likewise, grouping by the source IP and destination port and then looking for large numbers of different destination addresses reveals network scans. Expressed in SQL language, a search for network scans might look like:

```
SELECT source_ip_address, destination_port
COUNT(DISTINCT(destination_ip_address))
AS DST_IP
FROM my_log_table
GROUP BY source_ip_address, destination_port
HAVING DST_IP > 10
```

Examine the output with some Perl and you've got a list of potential scans. You may want to add in some code to adjust your sensitivity to different ports and address ranges. For example, some UDP/53 packets bouncing off of a firewall is probably not as important as TCP/22 packets, so you might want to require 50 of those packets before you get upset instead of 10. Then again, maybe it is more important; you should adjust the sensitivity in a manner appropriate to your situation.

There are several options for what to do with the list of IP addresses that appear to have scanned your equipment. One fairly extreme option would be to block all packets from that IP address to any of your systems. This would help keep intruders out and could be automated, but that could lead to a self-inflicted denial of service and so is

not a good idea. In deciding what to do, it is important to bear in mind that usually the IP address that the scan came from is not actually the IP address of the computer the scanner was sitting at. It is often another machine that has been compromised. In light of that, a good solution is to send a polite but firm email to the system administrator detailing what happened and to include some logs. This will usually solve the problem and can be safely scripted using a Perl WHOIS module to find administrator contacts for IP addresses (*whois.abuse.net* works well if you have a domain name). It is wise to design the script so that it errs on the side of caution when deciding whether to send mail. Crying wolf will not help your cause. If the recipient of the mail fixes the system, it will cut off one potential avenue of attack and have the added benefit of raising the bar for security in general. Even if nothing comes of the email, by tracking scan activity you will gain insight into what services may be exploitable.

The same general process can be applied to a host's authentication logs. Making effective use of the logs is simply a matter of defining a pattern of activity that appears unusual. For example, the use of `su` to elevate privileges may not be uncommon, but what about more than one failed attempt to become root without success? A few attempts followed closely by success might indicate a forgotten or mistyped password. However, without a successful attempt closely following, it starts to sound more suspicious. It is straightforward to write a Perl script that parses syslogs and checks for successful `su` attempts within, say, 15 seconds after an unsuccessful attempt. The script could then mail the administrator a list of suspicious usernames and times of attempted accesses.

Catching Misconfigurations

A major threat to the security of any system is misconfiguration – whether accidental or intentional. An administrator may configure a system in a way that puts it in an insecure state. Good logging can help detect this condition, hopefully before it becomes a problem. It can also help educate users and administrators by pointing out which commands can create security holes. The following example outlines how to detect someone misconfiguring a router.

Many routers support command-level accounting via the TACACS+ protocol. If command accounting is enabled, the TACACS+ accounting server will receive logs of every command entered (for a certain privilege level), what time it was entered, and which user entered it. These command logs can be parsed by a Perl script and, optionally, loaded into a database. Whether from the database or as it goes through the logs, the Perl script can then watch for commands such as `no access-list` or `no service password-encryption` that could create security problems. An alert could be paged or emailed to an administrator should such a command be entered. In a network with a small number of administrators, you could even make the script mail directly to the person who made the change (since you aren't all sharing the same passwords, right?).

Future Work

Simple scripts and databases are great for catching the easy stuff, but what about more stealthy attacks? Expanding the time span of your search will help catch some things, but this requires more and more processing power and storage space. One potential solution is to combine data from several different sources into one system. With data from application syslogs, firewall logs, and command accounting logs, for example, you could assign probabilities or levels of alert to different events from each. This would give a script a common ground for comparing events from different sources of

data. If three events, each with a low probability of occurrence, happen on the same system, there is a greater cause for alarm than if any occur individually. In this way, several events that by themselves would not be significant enough to cause alarm can be put together into a more complete picture. For example, a few TCP/22 packets denied to a few hosts on a subnet is perhaps nothing to be concerned about, but if you also notice a few unusual syslog messages from sshd on a machine on that subnet, and a few files that root owns get changed, then it would be a good idea to look at that machine. The more disparate sources of data are included, the more difficult it will be for attackers to slip through unnoticed.

HOSUS (honeypot surveillance system)

by Lance Spitzner

Lance is a security geek whose passion is using honeypots to study blackhats. This love for tactics began as a tanker in the US Army's Rapid Deployment Force. He is a senior security architect with Sun Microsystems.

lance@honeynet.org

Within the past several years, the information security community has increasingly recognized the value of honeypots. First discussed in 1989 and 1990 by Clifford Stoll[1] and Bill Cheswick,[2] honeypots are a unique security technology; they are resources designed to be attacked. Many people have different interpretations of what a honeypot is. For the purposes of this paper, I will use the following definition for honeypots: a security resource whose value lies in being probed, attacked, or compromised.[3]

This is a highly flexible definition, but then again, honeypots are a highly flexible technology, able to satisfy a variety of different goals and objectives. Commercial vendors, such as ManTrap, Smoke Detector, or Specter, have developed honeypots that can be used to detect attacks.[4] Other organizations, such as the Honeynet Project, have deployed honeypots for research purposes.[5] This paper attempts to describe one possible deployment of honeypots, called HOSUS (HOneypot SUrveillance System), a concept based on the Navy's SOSUS (SOund SUrveillance System) program.[6]

SOSUS

During the Cold War, one of the greatest threats facing the United States was the Soviet nuclear submarine threat. Submarines could move virtually undetected through the oceans. Used as platforms to launch nuclear attacks or gather intelligence based on intercepted signals, submarines could covertly collect sensitive information on or obliterate a country. One of the greatest assets of a submarine is its ability to operate clandestinely. Nothing is more frightening or more dangerous than an enemy you cannot find or track.

To counter this threat, the United States Navy in the 1950s developed and first deployed the SOSUS program. The intent was to monitor and track enemy submarines, thus neutralizing their greatest advantage and diminishing the threat. SOSUS consists of hydrophones placed along the bottom of various oceans. These hydrophones are linked together to passively capture activity-generated sounds, which are then used to identify, better understand, and track enemy threats.

HOSUS

Much as the United States faced hidden threats in the vastness of the oceans during the Cold War, organizations now face an even greater magnitude of hidden threats in the vastness of cyberspace. Just like the oceans, the Internet is an international domain, where threats come and go, allowing the enemy to strike at a time and target of their choosing. It is extremely difficult to identify and track this threat. HOSUS can provide a solution similar to the one provided by SOSUS. Like hydrophones that passively collect data from the ocean's depths, honeypots deployed throughout the Internet can passively capture attacker activity.

As an information collection and detection technology, honeypots have several advantages. First, they have no real production value, so any activity sent to them is most likely a probe, scan, or attack. This dramatically reduces false positives, one of the greatest challenges faced by most detection technologies. In addition, honeypots can also capture unknown attacks, reducing false negatives, as demonstrated with the Solaris dtspcd exploit captured in the wild in 2002.[7] Last, unlike most detection technologies, honeypots can interact with the attacker, giving more information on the

1. Clifford Stoll, *The Cuckoo's Egg* (New York: Doubleday), 1989.

2. Bill Cheswick, "An Evening with Berferd," *USENIX Proceedings*, January 20, 1990.

3. Lance Spitzner, *Honeypots: Tracking Hackers* (Boston: Addison-Wesley, 2002).

4. ManTrap: *http://www.mantrap.com*; Specter: *http://www.specter.com*; Smoke Detector: *http://palisadesys.com/products/smokedetector/prod_smokedet.shtml*.

5. The Honeynet Project: *http://www.honeynet.org*.

6. SOSUS: *http://www.pmel.noaa.gov/vents/acoustics/sosus.html*.

7. Solaris dtspcd exploit: *http://www.cert.org/advisories/CA-2002-01.html*.

attacker's activities and intent. Examples of this capability can be found in the series of Know Your Enemy white papers and challenges sponsored by the Honeynet Project.[8]

Threats using active measures, such as probes, attacks, or exploits, would be captured by these devices. Once correlated at a central point, this information can give organizations a much better understanding of the threats they face within cyberspace. More importantly, this information can potentially detect activity and predict attacks before they happen. However, HOSUS has the potential of capturing far more information then SOSUS ever could.

A concept similar to this has already been employed, though in a limited fashion. An organization known as the Honeynet Research Alliance,[9] an extension of the Honeynet Project, has passive surveillance devices, known as Honeynets, deployed throughout the world (as of July 1, 2002, they currently have 10 Honeynets). Data is passively collected on threats and attacks, then forwarded to a central point for data correlation and analysis (see Figure 1). This data has proven extremely valuable, resulting in analysis and publication of the characteristics and methodology of many different threats within cyberspace. The HOSUS concept could be employed on a much larger scale.

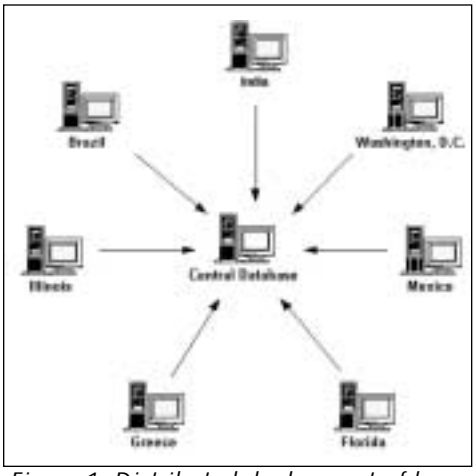


Figure 1: Distributed deployment of honeypots (in this case honeynets) passively collecting and then forwarding data to a central location. Source: Honeynet Research Alliance.

Deployment

There are two approaches to deploying the HOSUS concept: low interaction and high interaction. Honeypots are categorized by the level of interaction they provide to the attacker.[10] The greater the interaction, the more functionality honeypots have. For example, a low-interaction honeypot would emulate a Linux server running the wu-ftp service, limiting the amount of interaction the attacker would have with the system. A high-interaction honeypot would be a real Linux server running a real version of the wu-ftp service; there would be no limitation, since the attacker would have access to a real system. The attacker could exploit the service, take over and reprogram the computer, and then use it as a base for communication. The greater the level of interaction, the more we can learn about the attacker. However, the greater the interaction, the more work involved and the greater the risk the system could be subverted to attack or harm other non-honeypot systems.

Both low- and high-interaction solutions have their advantages with a HOSUS deployment. Low-interaction solutions are much simpler to deploy and maintain. But they are limited primarily to the transactional information of an attack, such as IP addresses, port numbers, and the time/date of the attack. Depending on the level of emulation with the low-interaction solution, some of the attacker's activities, such as login attempts, could be captured. This data can be extremely useful for detection, early warning, and prediction of activity, or statistical analysis of attack behavior.

High-interaction honeypots have the advantage of capturing far greater levels of information. They provide real operating systems and applications for attackers to interact with, just as they exist in the real world. One example of high-interaction honeypots, Honeynets, could be used to capture detailed information on the enemy, including their communications, latest tools and techniques, motives, and organization. Additional measures could be taken to create realistic Honeynets, perhaps even solutions that contain false information designed to mislead attackers. These Honeynets could be customized to appear as different targets, such as a university, government , or hospital site.

8. Know Your Enemy papers: *http://www.honeynet.org/papers/*; challenges: *http://www.honeynet.org/misc/chall.html.*

9. The Honeynet Research Alliance: *http://www.honeynet.org/alliance/*.

10. Spitzner, *Honeypots: Tracking Hackers*.

The quantity of passive listening devices deployed has a direct correlation to the amount of data collected and the value and statistical significance of the data analysis. The more sensors (honeypots) you can deploy, the more data you can obtain. To facilitate this wide deployment, it's possible to create rapidly deployable honeypot sensors. One idea is to create a simple appliance, such as a bootable CD-ROM. The CD-ROM would contain all the required software for the establishment and maintenance of the honeypot. It would be preconfigured to remotely and securely log all captured information to a central collection point. To facilitate ease of deployment, honeypots could also be pre-configured to passively monitor any IP address that is not specifically assigned to a system. This allows for easy and rapid deployment within most organizations. Whenever the honeypot sees activity for unassigned IPs, it simply assumes the identity of the victim, interacts with the attackers, prevents outbound attacks, captures the information, then securely forwards that information to the central data collection point.

The idea of monitoring unused IP space is not new, having been demonstrated by organizations such as CAIDA[11] and Arbor Networks, Inc.[12] However, in addition to monitoring IPs in unused networks, HOSUS monitors unused IPs within production networks of valid organizations as well. And honeypots take this concept one step further by not only monitoring but also interacting with attacks.

For a low-interaction deployment, technology like this already exists, such as the open source solution honeyd,[13] developed by Niels Provos. Honeyd is a low-interaction solution that emulates a variety of operating systems and services. When combined with a technology called arpd, honeyd can dynamically monitor unused IP space, then interact with any activity or attacks bound for those systems. A high-interaction solution would be more difficult to automate the deployment process, but would also have far greater information-gathering capabilities. It may be possible to build a Honeynet solution that also boots off a single CD-ROM, creating the Honeynet architecture, fulfilling data control, data capture, and data collection requirements. Then the Honeynet would only need to be populated with target systems. This process could even be streamlined further by creating virtual Honeynets,[14] multiple systems running off a single physical computer. Similar to honeyd, virtual Honeynets already exist and have been successfully deployed.

Risk

Just like any technology, HOSUS has inherent risks. The greatest risk is identification by the enemy. If the enemy can identify the existence and location of the deployed honeypots, he can neutralize their effectiveness. In the case of the low-interaction honeypots, the attacker can merely avoid the devices, avoiding detection. With high-interaction solutions, the attackers could not only avoid the systems but, if they so chose, feed it bad information, establishing, for example, a false IRC channel with bogus communications. A second risk exists: the honeypots can potentially be compromised and then be used to attack or harm other non-honeypot systems. This risk is especially prevalent with high-interaction honeypots, as we provide actual operating systems for the attackers to interact with.

These risks can be mitigated. By making the deployment of low-interaction honeypots simple and efficient, their identity and location can quickly be changed with minimal impact. Honeypots can be rotated to new locations on a weekly or monthly basis. In cyberspace, unlike the ocean, it is extremely easy to reconfigure and redeploy assets.

11. CAIDA, "Inferring Internet Denial-of-Service Activity," *http://www.caida.org/outreach/papers/2001/BackScatter/*.

12. Arbor Networks, "A Snapshot of Global Worm Activity," *http://research.arbor.net/up_media/up_files/snapshot_worm_activity.pdf*.

13. Honeyd: *http://www.citi.umich.edu/u/provos/honeyd/*.

14. Virtual Honeynets: *http://www.honeynet.org/papers/virtual/*.

This capability also exists with Honeynet technology. Honeynets can mitigate this risk even further by creating a highly realistic environment, running services and applications just as they would be found in a production environment.

For the risk of compromise, measures can be taken to control the attacker. For low-interaction honeypots, the emulated services are created to limit the attacker's interaction. These emulated services are designed not to be compromised; they do not provide enough functionality for the attacker to exploit. At most, they merely appear as vulnerable or exploited services. For high-interaction solutions, data-control mechanisms can be used to control outbound connections, mitigating the risk of the honeypot harming others. One example is Hogwash,[15] a solution that allows hostile egress from the honeypot but alters a bit in the malware to negate the outbound attack. This provides the maximum realism for an intruder inside the honeypot, leading the miscreant to believe the attack tool to be flawed. Other examples of data control include data throttling and counting outbound connections.

Conclusion

The purpose of this paper is to highlight the value of the honeypot deployment concept HOSUS. Similar to the hydrophones deployed during the Cold War, distributed honeypots could be used to passively collect information on threats and hostile activity within cyberspace. Once centrally correlated, this information could then be analyzed to better understand the threats that exist, detect indications of hostile activity, and prevent or, if required, defend against the cyberattack. The types of deployment, low interaction and high interaction, each has its advantages and disadvantages, depending on the data to be captured. Most likely, a successful deployment would require a combination of both technologies. However, both technologies share the same risks: detection and compromise. HOSUS is one possible method to better understand and protect against cyberthreats. If you are interested in learning more about honeypot technologies, *http://www.tracking-hackers.com* is an excellent place to start.

15. Hogwash: *http://hogwash.sourceforge.net*.

the case for network infrastructure security

by George M. Jones

George Jones is a network security architect for UUNET. In previous lives he worked at BankOne, CompuServe, and Ohio State University. He has been noodling around with Emacs since '79, UNIX since '85, and security things since '97. What a long strange trip it's been.

gmj@pobox.com

"The network is the computer." – Sun Microsystems, ca. 1984

"The network is the network, the computer is the computer. Sorry about the confusion." – Anonymous

What Does "Network Security" Mean?

What do you mean when you say "network security"? Firewalls? Intrusion detection systems? Anti-virus software? Authentication systems? System and application hardening? These are all fine and even necessary elements of "current best practice" if you're running a small office network or even a medium-sized corporate intranet.

But what if you're running a global Internet backbone with over 4700 routers, announcing over 60% of all routes, and have over 600 routers and switches in 25 hosting data centers? What does "network security" mean when you *are* the network and have no perimeter?

Chances are that in your world, you are closer to the first scenario. It is the premise of this article that while many of the solutions for smaller networks don't scale, many of the problems in the larger networks do apply generally, and that ignoring them may result in widespread disruption of service.

The main goal for large networks is availability. The bits should keep flowing, preferably to the right place. While integrity and confidentiality are important problems, it is assumed that these are handled "at the end of the pipe" by VPNs, host-based controls, good security policy and practice, etc. Assuring availability is a larger problem than it might at first seem. Let's take a look at some (relatively) recent problems.

Some Real Problems

DDoS Attacks

The greatest foreshadowing (not counting the Morris Worm of '88) of what could go wrong occurred in February 2000. Distributed Denial of Service (DDoS) attacks were launched, disabling Yahoo, eBay, Amazon.com, and others [lem01]. These attacks were made possible because hackers were able to "own" many poorly protected hosts across the Internet. Since then, DDoS defense has been a hot topic with governments, researchers, standards bodies, and network operators. A few products have even come on the market to address the problem. Still, there are no generally accepted solutions, technical or social, that adequately address the issue. In 2001, Code Red and Nimda demonstrated the speed with which worms can spread (and that we clearly have not solved the problem of insecure hosts). A paper [sta02] presented at the 2002 USENIX Security Symposium, titled "How to Own the Internet in Your Spare Time," demonstrates the current magnitude of the problem and suggests solutions, both technical and social.

The bottom line: DDoS attacks have already caused some disruption of service and have the potential to do far greater damage. Solutions are needed.

Bugs That Enable Hacking of the Network

Here is a sampling of some "recent" bugs that enable hacking of network infrastructure.

SNMP VULNERABILITIES

In February 2002, CERT announced vulnerabilities [cer02] that had been discovered in SNMP trap and request handling.[1] SNMP is the most widely used protocol for managing and monitoring network infrastructure. The root of the problem was our old friend the buffer overflow. In this case it was in the ASN.1 encoding/decoding that underlies SNMP. There are apparently very few ASN.1 implementations, so the result was that most SNMP implementations were vulnerable. A single malformed packet could cause the device to crash, or at least disable the IP stack. Imagine your core routers all suddenly rebooting.

NTP BUGS

In April 2001, a buffer overflow was discovered in certain UNIX-based NTP (Network Time Protocol) daemons [sec01]. Exploit code was published. Cisco published their own advisory in May 2002 [cis02]. The Cisco advisory stated that they were unable to reproduce the problem. Sources known to this author were able to exploit the vulnerability in IOS. Moral #1: Bugs can be "out there" a long time before they are exploited. Moral #2: Don't believe everything you read. Moral #3: Trust, but verify.

TELNET BUFFER OVERFLOWS

Going back a little further, to December 2000, we see that all problems are not related to buffer overflows in routers. A memory leak in the Telnet process on catalyst switches [cis00] caused them to reboot. Bye-bye desktops. Bye-bye Web servers . . .

SSH VULNERABILITIES

But you are being good. You're not using Telnet with its clear-text passwords. You use SSH to manage your devices. Have we got a vulnerability or two for you . . .[cis01]

Conflicting Priorities

Commercial vendors and network operators have conflicting priorities.[2] Vendors are interested in selling new equipment and software. Network operators are interested in operating networks. Vendors tend to focus effort on developing new products (which invariably have new bugs). Network operators focus on operating networks. Vendors tend to view bug fixing as a distraction from new product development and sales. Network operators view bug fixes in existing products as essential to operating networks. Vendors tend to see their job as done when the bug is fixed in the latest release. Network operators see their job as done when the bug fix is deployed across all devices (including old/obsolete ones) in their operating networks.

Operations

Operational realities can adversely affect security, even if technical solutions are known and available.

ANTIQUATED CODE REVS REQUIRED

It is sometimes the case that antiquated code revs are required for production. This may be true, for instance, if the vendor has produced a "special" to address unique

1. SNMP has been rumored to stand for "Security Not My Problem."

2. The author acknowledges that this may be "unfair," since it portrays an external view of vendors as seen from the engineering/security trenches.

needs of a particular customer or if the customer has policies against running "bleeding edge" releases (as is the case in industries such as utilities and banking).

UPGRADES NOT POSSIBLE

Sometimes, in the real world, it is just not possible to upgrade code quickly. Maybe "the network guy" quit or was laid off, and consultants are not immediately available. Maybe the IT department has other priorities (i.e., the risk of a bug in networking infrastructure is not perceived as high). Maybe the vendor has not produced a fix for the bug in question, or it is not available on the old hardware that is still doing just fine at meeting the business needs it was purchased to address. "If it ain't broke . . ."

PEOPLE/REAL-WORLD ISSUES

This does not touch the non-technical yet very real issues of staffing and training. At the present time the entire telecommunications sector is experiencing serious financial difficulty, with all the attendant impact on priorities and funding.

Configuration

In defense of vendors, it can be argued that the majority of vulnerabilities in networking infrastructure are due to incomplete configuration or misconfiguration. Vendors are supplying the right knobs. They just need to be set correctly. That's where tools such as the Router Audit Tool [jon02a] come in.

SURVEY SAYS . . .

A quick survey of 471 routers across the Internet that showed up in traceroutes to 94 Web servers showed that 81 of them (17%) accepted connections from arbitrary sources on either port 22 (SSH), 23 (Telnet) or 80 (HTTP). This is bad – no filters on administrative access. Of those, 38 (8%) were listening on port 22, 57 (12%) were listening on 23, and thankfully only 5 (1%) answered on 80.[3]

Some Potential Problems

We have seen a sampling of things that are problems today. Now, let's take a look at Network Nightmares: The Next Generation.

Same Old Same Old

If past performance can be used to predict future behavior, we can project that:

- Vendors will continue to release new products.
- These products will, with non-zero probability, have bugs.
- Consumers will buy and deploy these buggy products.
- The numbers of deployed networks,[4] systems, and users will continue to increase.
- The product of the probability of bugs and the number of deployed systems will increase. In absolute terms, there will be more vulnerabilities.
- The number of trained (and employed) systems and network security administrations will not increase at the same rate. The result will be more misconfigured or unconfigured systems.

More Spew

The eBay/Yahoo attacks of February 2000 [lem01] may only have been the tip of the iceberg, as demonstrated in Staniford et al. [sta02]. Real and crippling DDoS attacks on major sites and networks are a distinct possibility.

3. Thanks to Pete White for suggesting this survey method.

4. From 1/5/00 to 3/5/02, the number of advertised routes grew from 76,182 to 110,842, a 45% increase, even in the face of a down economy. Source: *http://www.employees.org/~tbates/cidr-report.html*.

References and Resources

References

[ahm00] "Network Infrastructure Insecurity," Rauch Ahmad: *http://www.blackhat.com/presentations/bh-asia-00/jeremy-dave/jeremy-dave-asia-00-network.ppt*.

[cer02]"Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)," CERT/CC: *http://www.cert.org/advisories/CA-2002-03.html*.

[cis00] "Cisco Catalyst Memory Leak Vulnerability," Cisco Systems: *http://www.cisco.com/warp/public/707/catalyst-memleak-pub.shtml*.

[cis01] "Multiple SSH Vulnerabilities," Cisco Systems: *http://www.cisco.com/warp/public/707/catalyst-memleak-pub.shtml*.

[cis02] "Cisco Security Advisory: NTP Vulnerability," Cisco Systems: *http://www.cisco.com/warp/public/707/NTP-pub.shtml*.

[eff02] EFF Homepage, Electronic Frontier Foundation: *http://www.eff.org/*.

[jon02a] "The Router Audit Tool and Benchmark," George M. Jones et al., Center for Internet Security: *http://www.cisecurity.org*.

[jon02b] "Network Security Requirements for Devices Implementing Internet Protocol," George M. Jones, editor: *http://www.port111.com/docs/netsec-reqs.html*.

[lem01] "DDoS Attacks – One Year Later," Robert Lemos: *http://zdnet.com.com/2100-11-527990.html?legacy=zdnn*.

[sec01] "Ntpd Remote Buffer Overflow Vulnerability," SecurityFocus: *http://online.securityfocus.com/bid/2540/info/*.

[sta02] "How to 0wn the Internet in Your Spare Time," Stuart Staniford, Vern Paxon, and Nicholas Weaver: *http://www.icir.org/vern/papers/cdc-usenix-sec02/*.

[yas01] "Latest Hacker Target: Routers," Rutrell Yasin: *http://www.internetweek.com/story/INW20011217S0004*.

[yro02] "Your Rights Online," Slashdot: *http://www.slashdot.org/yro/*.

Do You Know Where Your Routes Are?

Attacks on routing infrastructure (routers, routing protocols) have been a matter of great speculation for some time [ahm00, yas01]. Should they materialize, they could result in denial of service or misrouted traffic on a large scale. Does your intrusion detection system alert you when someone advertises a more specific route for your address blocks, or when someone logged in to your router, set up a tunnel, and policy routed all traffic for a single customer down the tunnel?

Sed Quis Custodiet Ipsos Custodes?

Juvenal asked, "But who watches the watchmen?" This is a question we need to ask again as we design and legislate confidentiality out of our systems. Networks provide aggregation points that are natural targets for those who would practice surveillance. See [eff02] and [yro02] for a list of current abuses and privacy threats mixed with some strong opinions and wild ravings.

Short-Term Solutions

What can you do *today* to improve the security of your network?

- Be vigilant. Be clued. The first and best line of defense for any network is network/security admins doing things like: staying on top of current vulnerabilities, being aware of current best practices and tools, implementing fixes as needed, watching logs, etc.
- Patch, patch, patch. Vendors routinely put out patched versions of code to fix newly discovered problems. Running old/unpatched code is inviting trouble. Check your vendor advisories.
- Harden network infrastructure using current best practices and tools. See the resources section for some suggestions.

Medium-Term Solutions

What can you do in "medium" term to improve the security of your network?

- Policy. You do have a policy even though you may not have written it down. Why does your network exist? Who can use it and for what? Who manages it? How are changes made? Having clear, documented answers to these questions backed by those in charge of the organization provides a foundation for all other standards, requirements, practices, etc.
- Standards and requirements. What features do you need to be able to implement your policy? Does your current infrastructure support them? Can you clearly communicate them to your vendors? Some areas to consider (addressed more fully by the author in [jon02b]) include:
  - device management
  - user interface
  - IP stack (RFC compliance, disable services/ports, DoS tracking, traffic monitoring/sampling, rate limiting . . .)
  - packet filtering
  - event logging
  - authentication, authorization, and accounting (AAA)
  - layer-2 issues (VLAN isolation).
- Industry participation. Work with others to define the important problems and generate solutions. This could range from policy and legislative work, to generat-

ing industry-wide consensus on required security features, helping to define standards and best practices, and developing tools to ensure their application.

Long-Term Solutions

What can be done in the long term to improve the security of your network?

- Cooperation and communication. Staniford et al. [sta02] suggest a network analog to the Center for Disease Control (CDC) for network issues. There have been some attempts to form such an organization (e.g., www.first.org). The big question is how to ensure participation.
- Consensus. Consensus must be achieved about what the "right" solutions are.
- Conformance. Vendors must be convinced to conform to the consensus solutions. The strongest incentive for conformance would be to have many customers making conformance a condition of purchase in contracts.
- Compliance certification. Once vendors are convinced and implement the requested features, there will be a need for testing and certification. It is likely that larger organizations will do this "in house," while smaller organizations will need to rely on some external testing entity.
- Coercion ("Send lawyers, guns, and money"). In the end (maybe sooner than we would think/like), we will have lawyers, legislators, insurance companies, and auditors telling us how to run and secure networks.

The Big Question

The big question, assuming this assessment of the problem is correct, is whether anything will be done before we have a major network outage. Can we, as a community, proactively address the issues raised here, or will it take a major disruption of services for "Network Security" to be recognized as an important priority? Time will tell.

Resources for Securing Cisco Routers

"Hardening Cisco Routers," Thomas Akin: *http://www.oreilly.com/catalog/hardcisco/*.

"Improving Security on Cisco Routers," Cisco Systems: *http://www.cisco.com/warp/public/707/21.html*.

"The Router Audit Tool and Benchmark," George M. Jones et al., Center for Internet Security: *http://www.cisecurity.org*.

"Securing Cisco Routers Step-by-Step," John Stewart and Joshua Wright: *http://www.sansstore.org* (forthcoming).

Rob Thomas' Security Articles, Rob Thomas: *http://www.cymru.com/~robt/Docs/Articles/* (articles/guides to securing IOS, JunOS, BGP, DoS tracking, etc.).

Thanks to the whole UUNET net-sec team (past and present) for feedback.

security threats to IP telephony-based networks

by Ofir Arkin

Ofir Arkin is the founder of the Sys-Security Group, a non-biased computer security research and consultancy body. In his free time he enjoys doing computer security research. His publications and work is available from the group's Web site, *http://www.sys-security.com.*

ofir@sys-security.com

Introduction

Privacy and security are mandatory requirements with any telephony-based network. Although not perfect, over the years a certain level of security has been achieved with traditional telephony-based networks.

On the other hand, IP telephony-based networks, which might be a core part of our telephony infrastructure in the near future, introduces caveats and security concerns which traditional telephony-based networks do not have to deal with, long ago forgot about, or learned to cope with.

Unfortunately, the risk factors associated with IP telephony-based networks are far greater than traditional telephony-based networks.

The security concerns associated with IP telephony-based networks are overshadowed by the technological hype and the way IP telephony equipment manufacturers push the technology to the masses. In some cases IP telephony-based equipment is being shipped although the manufacturer is well aware of the clear and present danger to the privacy and security of the IP telephony-based network its equipment is part of.

This article highlights the security risk factors associated with IP telephony-based networks and compares them, when appropriate, with the public switched telephony network (PSTN) and other traditional telephony-based solutions.

What Is IP Telephony?

IP telephony is a technology in which IP networks are being used as the medium to transmit voice traffic.

IP telephony has numerous deployment scenarios and architectures which the following terms are usually associated with:

- *Voice over IP* (*VoIP*) – describes an IP telephony deployment where the IP network used as the medium to transmit voice traffic is a managed IP network.
- *Voice on the Net* (*VON*) or *Internet telephony* – describes an IP telephony deployment where the IP network used as the medium to transmit voice traffic is the Internet.

With any IP telephony-based deployment scenario, the underlying IP network will carry data as well as voice. The term *Converged Network* is used to describe networks which carry both voice and data. This is in contrast with the current *Public Switched Telephone Network* (*PSTN*), where voice and data are being carried on physically separated networks.

Different protocols play different roles in IP telephony. With any IP telephony-based network, several types of protocols will be responsible for different aspects of a "call":

Signaling protocols perform session management and are responsible for:

- Locating a user – the ability to locate the called party.
- Session establishment – the ability to determine the availability of the called party as well as its willingness to participate in a call. The called party is able to accept a call, reject a call, or redirect the call to another location or service.

- Session setup negotiation – the ability of the communicating parties to negotiate the set of parameters to be used during the session, including, but not limited to, type of media, codec, sampling rate, etc.
- Modifying a session – the ability to change session parameters during the call, such as the audio encoding, adding and/or removing a call participant, etc.
- Tearing down a session – the ability to end a session.

Media transport protocols are responsible for the digitization, encoding (and decoding), packing, packeting, reception, and ordering of voice and voice samples.

IP telephony-based networks also make use of other protocols and technologies which are common to any IP-based network, such as DNS and quality of service,

A Generic Call-Setup Process

When a user places a call on an IP telephony-based network, the signaling protocol its IP phone supports will locate the called party either directly or by using other servers on the network, determine the called party's availability and willingness to participate in a call, and negotiate the parameters to be used during the call.

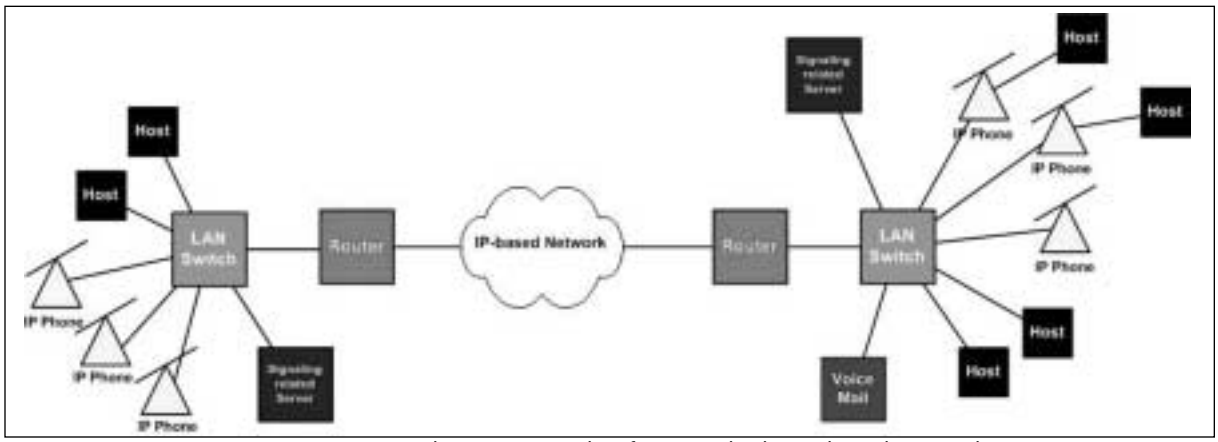


Figure 1: A very abstract example of an IP telephony-based network

The actual voice samples are carried by a media transport protocol, such as the Realtime Transport Protocol (RTP), which samples human speech according to the parameters negotiated by the signaling protocol during the call-setup process. Some, but not all, of the media protocol's operations will be controlled by the signaling protocol.

During the call, when needed, the signaling protocol is used to change a call parameter. It is also responsible for tearing down the call.

The signaling information might traverse several signaling-related servers, while the voice samples are being sent directly between call participants.

Parameters other than security and privacy, must be taken into account when designing an IP telephony-based solution. They include but are not limited to:

- Availability
- Speech quality
- Quality of service
- Scalability

Although these parameters do not seem to be linked with security, the ability of a malicious party to interfere with the operation of the network will pose a direct threat to its availability and therefore will downgrade its role as critical infrastructure.

1. A phreaker is one who engages in phreaking, cracking phone systems.

2. A softphone is telephony-based software running on a PC.

Why IP Telephony Is at Risk

IP telephony brings the terms "phreaker"[1] and "hacker" closer together than ever before. Several characteristics of IP telephony make it easier for a hacker to try to compromise and/or control different aspects or parts of the IP telephony-based network.

Compared to the PSTN, IP telephony-based networks face a greater security threat as a result of a combination of key factors outlined below.

Use of the IP Protocol

Since IP telephony is using the IP protocol as the vessel for carrying both data and voice, it inherits the known (and unknown) security weaknesses that are associated with the IP protocol.

IP Networks Are Common

IP networks are easily accessible, allowing more people to explore security issues, and for security vulnerabilities when found to be published or otherwise disseminated. This is unlike the obscurity which characterizes the PSTN.

Signaling and Media Share the Same Network

Although they might take different routes, signaling information and media (voice samples), with IP telephony-based networks, share the same medium: the IP network. Unlike the PSTN, where the only part of the telephony network the signaling and media share is the connection between the subscriber's phone and its telephony switch (thereafter the signaling information will be carried on a different network physically separated from the media – the SS#7 network), with IP telephony no such isolation or physical separation between voice samples and signaling information is available, increasing the risk of misuse.

The Placement of Intelligence

With the PSTN the phones are no more than a "dumb terminal" where the telephony switch holds the actual intelligence. With some IP telephony-signaling protocols (e.g., the Session Initiation Protocol – SIP), some or all of the intelligence is located at the endpoints (IP Phones, softphones,[2] etc.). An endpoint supporting this type of signaling protocol will have the appropriate functionality and ability to interact with different IP telephony components and services as well as different networking components within the IP telephony-based network. A malicious party using such an endpoint, or a modified client, will have the same ability to interact with these components. This is in contrast to the PSTN, where a phone is only able to interact with its telephony switch.

The ability of an endpoint to interact with more IP telephony-based elements and network components poses a greater risk of misuse for an IP telephony-based network compared with the PSTN, where the switch a phone is connected to is the most likely to be attacked.

No Single Authority (Entity) Controls an IP Medium (the Network)

With several IP telephony architectures, the signaling and media information will traverse several IP networks controlled by different entities (e.g., Internet telephony, different service providers, different telecom companies). In some cases, it will not be possible to validate the level of security (and even trust) that different providers will

enforce with their network infrastructure, making those networks a potential risk factor and an attack venue.

The Nature of Speech

Without adequate speech quality, subscribers/users will avoid using IP telephony solutions. Speech quality with IP telephony is a function of several key factors, such as latency (delay), jitter (delay variation), and packet loss. With the PSTN some of these factors were long ago dealt with or are a non-issue.

A good example is jitter. During a call setup with the PSTN, a dedicated communication path between several telephony switches, also known as a trunk, is set, allowing a voice passage between the call participants. Since this is a dedicated communication path, voice traffic between the call participants will take the same route during a call. Therefore jitter is less likely to occur.

The number of factors affecting speech quality, and the ways to stimulate those conditions, are far greater with IP telephony-based networks than with the PSTN.

Unacceptable speech quality is an availability problem, which jeopardizes the critical infrastructure tag IP telephony has.

IP Telephony Infrastructure

The IP telephony infrastructure is usually put together from standard computer equipment and in many cases is built upon known operating systems, which are fully functional. The IP telephony infrastructure components interact with other computer systems on the IP network. They are thus more susceptible to a security breach than the equipment combining the PSTN, which is usually proprietary equipment whose operation is somewhat obscure.

Components of the IP Network

Networking components and the other computer equipment (e.g., network servers) combining to make up the IP network that serves the IP telephony infrastructure are the same common components found in many other IP networks. They offer other attack venues.

IP Telephony Protocols

IP telephony-related protocols were not designed with security as their first priority or as a prime design goal. Some of those protocols added security features when newer protocol versions were introduced. Other IP telephony protocols introduced some security mechanisms only after the IETF threatened not to accept a newer version of the protocol if security was not part of it. Despite such demands and an effort to introduce "decent" security mechanisms within some IP telephony protocols during their design phase, in some cases inappropriate security concepts were adopted only to satisfy the IETF. Some of those security mechanisms were simply not enough, regarded as useless or impractical, giving a false sense of security to the users of these IP telephony protocols.

An example of a security technology that might cause more harm than good is encryption. Encryption affects voice quality, since it adds delay on top of the usual delay experienced with an IP telephony-based network. Although some IP telephony-related protocol specifications mandate the use of encryption, it is sometimes simply not feasible to use encryption with those protocols. An example is the draft version of

IP telephony-related protocols were not designed with security as their first priority or as a prime design goal.

the new RTP protocol, which mandates the use of triple-DES encryption. We should not forget that most IP phones today are not powerful enough to handle encryption.

VPN technology is another good example of a security-related technology that degrades voice quality. Where we have more than two or three encrypted IP telephony "tunnels," voice quality is usually unbearable, the result of current encryption technologies combined with realtime multimedia demands.

Some security mechanisms offered by different IP telephony protocols might break the protocol functionality and even the functionality related to an IP telephony-based network.

IP telephony protocols are open to malicious attack to the degree that the attacker would be able to compromise and/or control different aspects or parts of the IP telephony-based network. The PSTN enjoys some level of obscurity in relation to security, the kind of obscurity which is not possible for a set of protocols using an openly developed IP telephony solution.

The fact is that IP telephony-based protocols are still going through several development cycles. The requirements for privacy and security are not being correctly balanced with what is feasible.

Supporting Protocols and Technologies

IP telephony protocols pose a threat to the security of IP telephony-based networks, but so do the supporting protocols and technologies that are usually part of an IP network; among these, we can name application protocols (e.g., DNS, quality of service) and internetworking technologies (e.g., Ethernet, Token Ring), and the list is long. Taking advantage of a supporting protocol or a technology being used in the IP network serving the IP telephony-based components might allow a malicious party to control different aspects or parts of the IP telephony-based network.

Physical Access

With IP telephony, physical access to the network or to some network component(s) is usually regarded as an end-of-game scenario, a potential for total compromise. A malicious party gaining physical access to the network or to a network element will have several key advantages over one having a similar physical access to PSTN equipment. This is a direct result of the way IP networks work, the placement of intelligence in some IP telephony-based networks, and the boundaries of physical security and access with the PSTN.

For example, if a malicious party is able to gain unauthorized physical access to the wire connecting a subscriber's IP phone to its network switch, the attacker will be able to place calls at the expense of the legitimate subscriber while continuing to let the subscriber place calls at the same time. With the PSTN, a similar scenario would unveil the malicious party when the legitimate subscriber took the handset off hook.

Design Flaws with IP Telephony Protocols

The IP telephony-related protocols contain several design flaws – not easily identified, but costly – that would allow an attacker to cripple an IP telephony network. One such flaw is a signaling protocol that does not maintain knowledge of changes made to the media path during a call. If one is able to abuse the media path, the signaling path will remain unnotified and clueless about the changes performed to the media path.

Another example is a signaling protocol that does not have an integrity-checking mechanism.

AVAILABILITY, OR LOW-TECH IS VERY DANGEROUS

IP telephony-based networks face a serious risk of availability. The availability risk does not result only from availability-based attacks against protocols, endpoints, network servers, and/or the kind of attacks designed to reduce the quality of speech or that target simple equipment malfunction(s). The main risk, and one that is even more basic, is the lack of electricity to power endpoints (e.g., IP phones) and other elements making up an IP telephony-based network or infrastructure.

NO ELECTRICITY? – NO SERVICE!

The electricity availability problem may strike anywhere along the path from one subscriber to another, anywhere on the network. While service providers would have to have redundancy and means to solve power-down problems as part of their license terms (at least in most Western countries), for a corporation, a small-to-medium business, or an individual subscriber this problem is more critical.

For a business the question of redundancy and power down means additional cost and economic burden, but for the subscriber at home it might mean life and death.

For a subscriber the phone is the critical infrastructure. Whenever things go wrong, the first thing most people do is to use their phone to get help. An IP phone depends on power. With most IP phones, power can be drawn either from a direct connection with an electricity outlet, or if the network infrastructure and IP phone supports it, from the LAN (power-over-LAN). If electricity is cut either to the subscriber's house (or any other location an IP phone is being used at) or to the network switch the subscriber's IP phone is connected to, the IP phone is useless.

For a subscriber, an IP phone simply cannot be depended upon as a critical infrastructure component if no electricity backup solution is available.

REDUNDANCY

If one IP telephony element fails within an IP telephony-based network and there is no redundancy, there is no availability either.

It all comes down to the economic burden of supporting availability in an IP telephony-based network.

Taking into account the other availability risks and targets within IP telephony-based networks, availability becomes one of the biggest concerns.

DIFFERENT IP TELEPHONY ARCHITECTURES

Although sharing the same basic threats, various deployment scenarios and IP telephony architectures differ from each other by the overall risk factor presented and the attack venues a malicious party might use. Securing IP telephony-based solutions is more complicated and challenging than securing the PSTN, where the major security issue is fraud.

IMPROPER IP TELEPHONY NETWORK DESIGNS

The currently offered network designs for the implementation of IP telephony-based networks do not provide proper mechanisms to defeat several basic hazards to the IP telephony network.

We can name a couple of examples:

IP telephony equipment (devices) is not being authenticated to the network, and this makes the work of the phreaker easier; in some cases, by plugging a rogue device to the network, free phone calls can be made.

In many IP telephony-based networks an IP phone's (that is, the user's) actual location is not checked against the credentials it uses. It is not enough that the network switch is able to perform "port security" and bind the port connected to an IP phone with the phone's MAC address. There should be a mechanism to correlate between the credentials presented, the MAC address the phone is using, and the physical port on the network switch it is connected to.

Non-Trusted Identities

Without the proper network design and configuration of an IP telephony-based network, one cannot trust the identity of another call participant. The user's identity, the "call-ID" information (i.e., a phone number or other means to identify a subscriber in IP telephony-based networks), is easily spoofed using any one of a variety of scenarios. An identity-related attack might occur anywhere along the path signaling information is taking between call participants.

A malicious party might use designated software to perform digital impersonation, adding to the attacker's arsenal of available tools, when spoofing an identity of a call participant or a targeted call participant, where the voice samples might have been gleaned from the IP telephony-based network itself.

Unlike IP telephony-based networks, spoofing identities with the PSTN is a much harder task to perform, and is usually performed only at the endpoints, where someone other than the intended subscriber answers the subscriber's phone, for example, or a calling party claims to be someone he/she is not.

What Is at Risk?

Everything is at risk. With IP telephony there is even greater meaning to the phrase that the security of a particular architecture is only as good as its weakest link. Multiple venues exist for a malicious party to choose from in order to launch an attack against an IP telephony-based network. Most disturbingly, in most cases it is only a question of subverting one network server or one IP telephony element (e.g., IP phones)[3] to achieve complete control over an IP telephony-based network or its functionality.

Conclusion

Each and every potential security threat examined within this article has shown that IP telephony-based networks face a greater risk of being breached than the Public Switched Telephone Network. Unfortunately, mitigating the risks highlighted within this article is not simple.

When examining the current IP telephony-based protocols and network designs, it is clear that both need to undergo major changes.

3. For more information, please see *http://www.sys-security.com/html/projects/VoIP.html*.

the kernelized secure operating system (KSOS)

Last August I had the pleasure of attending the USENIX Security Symposium in San Francisco. As a security practitioner and former OS designer, I've always considered this one of my favorite conferences, and I haven't missed one in quite a while. But as I attended the panel discussion of Microsoft's Palladium and the TCPA, I was once again struck with a curious sense of déjà vu. Palladium in particular seems to be an attempt to create a "trusted" hardware system to overcome the inability to create large monolithic operating systems with acceptable software quality.

It seemed to me that both Palladium and TCPA were rediscovering older security solutions to solve some of the same old problems. Much as I felt when Microsoft (and some of the open source OSes) announced the "invention" of symmetrical multiprocessing (SMP), I felt that someone, somewhere, had not done their homework. I felt the same way about VAX/VMS (and later, Windows NT) access control lists (ACLs), which were pale imitations of the Multics[1] ACLs.

There were SMP mainframes at least as far back as the 1960s in General Electric's GECOS[2] and, later, Multics operating systems – up to six processors in some cases. For those of you who don't remember, when Bell Labs pulled out of the GE/MIT Multics project (Project MAC), UNIX was conceived in part as a smaller, more practical implementation of the "Multics vision." IBM and (I think) UNIVAC also sold multiprocessor mainframes in the same era.

Now, I hate the "when I was a kid we had to feed wood into the boiler to power our steam computers" stories as much as everyone else. But, to be honest, the computer science and engineering disciplines do a terrible job of teaching the history of our field. And this leads to someone rediscovering the same old "new and novel" solutions to the same old problems about every five years, solutions that can often be found in the older literature.

Palladium and TCPA (and Linux and *BSD) continue to rediscover solutions that have been found before, in systems such as KVM/370,[3] Multics, SCOMP, and KSOS.

To understand KSOS and its place, you need to understand the time and background of its creation. It was the late 1970s and TOPS-10, TWENEX (later TOPS-20), UNIVAC EXEC 8, OS/360, and Multics were the denizens of this new thing called the ARPANET. The ARPANET backbone ran at 56Kbps and connected about 100 computers. There was also this new upstart thing that people were beginning to play with called UNIX, "6th Edition." It ran on these new 16-bit mini-computers from Digital Equipment, PDP-11s. Big ones, with 512K of core or semiconductor memory and 20MB washing machine drives. Programs were limited by the hardware to 64K bytes total or 64K instructions and 64K data, using split instruction and data addresses. SPLIT I/D was ugly. Don't ask. If you were there, I'm sorry to have reminded you.

There was a lot of interest in computer security. The government needed to process classified information, and computers were still expensive enough that they needed to be shared. There was lots of interest in allowing data at different classifications to be processed on the same computer at the same time, without "spilling" data across security levels.

by Tom Perrine

Tom Perrine is the Manager of Security Technologies at the San Diego Supercomputer Center, where his job description is "protect the privacy and intellectual property of our users." Involved with computer security since the '80s, he was a Multician, has testified to Congress concerning Carnivore, and studies computer security technology and how it relates to people and public policy.

tep@ARPA.NET

1. The best current online information about Multics (Multiplexed Information and Computing Service) can be found at *http://www.multicians.org/*.

2. The following online entry has most of the story right. The main mistake is the claim that Multics had no database and no transaction processing. Other than that, it's pretty on target: *http://wombat.doc.ic.ac.uk/foldoc/foldoc.cgi?GCOS*.

3. "KVM/370 in Retrospect," 1984 IEEE Symposium on Security and Privacy: *http://www.computer.org/proceedings/sp/0532/05320013abs.htm*.

The "Orange Book" hadn't been written yet. The experiences from developing Multics, SCOMP, and KSOS substantially influenced[4] the content of the Orange Book.[5] "Hackers" were still people who were curious about computers and wrote interesting code, mostly at MIT and Stanford AI Lab.

A few years earlier, in 1973, the seminal report on computer security – the "Anderson Report"[6] – had been published for the US Air Force. This report called for better software design practices, better programming languages, and something new called a "security kernel." It also suggested using formal mathematical models to prove that the kernel would operate correctly. This paper also, almost as an afterthought, described what we now call "automated intrusion detection" and noted that a primary way to compromise an operating system was to exploit "insufficient argument validation."

Yes, the Anderson Report described buffer overflows as a proven penetration method *and ways to avoid them* 30 years ago. We've obviously come a long way since then. So far that we need Palladium and TCPA.

The Anderson Report also cited an obscure little paper: "Notes on Structured Programming" by Edsger Dijkstra. This was about the same time that he wrote a letter to the *Journal of the ACM*, "Goto considered harmful." Strangely enough, it was the day of the Palladium/TCPA panel that we learned of his death. His two papers jump-started the entire structured programming movement of the 1970s and 1980s.

At about the same time (1973), a DoD-inspired mandatory formal security model was developed by Bell and LaPadula working at Mitre. This model formalized the DoD classification system into a set of mathematical rules called "mandatory access controls." The idea was that the site policy would override any "discretionary access controls" – that is, people could not give away data to unauthorized people. One rule ("simple security") prohibited data at a "higher" level from being read by a process at a "lower" level. Another rule ("*-property") prohibited a process at a "higher" level from writing to "lower" level data.

People started to design and develop "security kernels." These were small, well-defined cores upon which an OS could be written that would be small and "verifiable" using formal methods. This gets around the problem that verification methods and human minds weren't ready to deal with analyzing very complex systems. The idea was to concentrate all the security features, and only the security features, into a small kernel that would provide the base upon which a secure OS could be layered. This was imposing "least privilege" on the operating system itself, allowing the operating system to have bugs and yet not be able to compromise security. In 1976, Peter Neumann and others at SRI proposed "pSOS," a provably secure operating system.

In 1978, Neumann, John Nagle, and others at Ford Aerospace started work on a more ambitious project, which was actually expected to produce a running, practical, usable system, the Kernelized Secure Operating System (KSOS). Neumann went on to become the editor of *Risks Digest*, among many other projects. Nagle later worked in networking, producing Nagle's algorithm for merging tiny packets for TCP performance, which was published in RFC 896 and is part of every modern TCP/IP stack.

KSOS was intended to be a security kernel upon which a UNIX-like operating system could be built. It was recognized that the V6 UNIX kernel, at 10,000 lines of code[7] and 48(!) system calls, was much too big(!) to be formally specified or verified.

4. Jeff Makey, private communication. Jeff was an editor of the original Orange Book while at the National Computer Security Center.

5. Department of Defense, Trusted Computer System Evaluation Criteria: *http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html*.

6. J.P. Anderson, Computer Security Technology Planning Study, ESD-TR-73-51, ESD/AFSC, Hanscom AFB, Bedford, MA (Oct. 1972) [NTIS AD-758 206]: *http://seclab.cs.ucdavis.edu/projects/history/seminal.html.*

7. The "infamous" "Lions book," an annotated display of the UNIX V6 kernel for PDP-11, shows about 9000 lines of code, in 44 source files, including the .h files. Again available in print (after being quashed by AT&T in the seventies): John Lions, *Lion's Commentary on the UNIX 6th Edition with Source Code* (San Jose, CA: Peer-to-Peer Communications), 1996, ISBN 1-57398-013-7.

But by building a smaller security kernel that would implement the security features, a UNIX-compatible layer could be built on top of that "micro-kernel" that would provide a UNIX-compatible system-call interface. The micro-kernel approach lived on in later OSes such as MACH.

It was amusing to me that during the Palladium/TCPA panel, when asked about improving the quality of the operating system so that Palladium would be less necessary, the Microsoft speaker scoffed at "fixing millions of lines of code," implying that the OS was just too big to be written properly. Perhaps they need to "invent" least privilege and "micro-kernels" again!

But back to KSOS. First, the security kernel itself was designed and specified. The kernel was modeled as a finite-state machine, and the system calls were defined in terms of the state transitions that could occur. It was decided that by defining an initial known secure state, and then checking all possible state transitions (system calls) to make sure that they led to a known secure state, the kernel would be "verified." The design was documented and specified in the SPECIAL specification language. The kernel specification was considered manageable because there were only 32 kernel calls.

The kernel specification was examined using the Boyer-Moore theorem prover. The prover, which ran on a fairly large DEC 20, was eventually able, with *extensive* human assistance, to prove enough of the theorems to provide a significant level of assurance of "correctness." In this context, this means that the kernel specification contained no explicit violations of the Bell-LaPadula model, no "covert channels" or ways for data to be implicitly shared across mandatory access boundaries. Note that even with only 32 system calls, the specification had to be split into five pieces, which were separately verified. This verification allowed KSOS to be considered a candidate for an "A1" rating, "verified design," the highest Orange Book rating.

KSOS was implemented in a higher-level language, Modula-2. Think of the type safety of Pascal, with the package constructs of Ada or C++ (years before either language was designed). User programs could be written in Modula-2 or C. KSOS was not self-hosted; programs had to be compiled under V6 UNIX and copied to the KSOS system.

KSOS had some other interesting features for its time, such as multiple virtual terminals per real terminal (think of Linux virtual consoles), and a "trusted path" from the terminal into the security kernel. When a user hit the "secure attention" key (such as BREAK), all user programs were suspended and disconnected from the user's terminal. The terminal was connected straight to the kernel. This was to avoid applications from impersonating trusted applications or the kernel's authentication (login) or password change functions.

KSOS also implemented "shared memory segments," typed files, and what could be considered network firewall features – all before Berkeley completed 4.2BSD.

In 1981, KSOS development moved to Logicon in San Diego, where it was further enhanced and later served as the platform for several Navy and Air Force operational systems, such as ACCAT GUARD and USAFE GUARD. These systems used KSOS-hosted applications to provide multi-level secure application gateways on very secure DoD networks. The "final" version of KSOS for PDP-11 is described in Perrine, Codd, and Hardy,[8] which also includes some information about ACCAT GUARD.

8. Perrine, Codd, Hardy, "An Overview of the Kernelized Secure Operating System (KSOS)," Proceedings of the 7th DoD/NBS Computer Security Conference, September 24–26, 1984: *http://users.sdsc.edu/~tep/Presentations/Overview_Paper.text*.

Later, KSOS was ported to the VAX and became KSOS-32. That project was canceled, along with many other DoD computer security projects, in September 1988, shortly after the first user login was achieved.

Although KSOS (and SCOMP and Multics) made significant advances in computer security and software design methodologies and helped us to understand the problem of software quality and assurance, they have been mostly forgotten. These OSes, and their contemporaries, provided many features and services that are continually rediscovered or even "invented" every few years for new operating systems. Palladium and TCPA are just the most recent efforts to cover the same ground. In Orange Book terms, they are trying to go "beyond A1" into "trusted hardware," without first getting to B-level software architecture.

It may be that UNIX came along and swept up a new generation, and the "old skool" operating systems and their "old guard" were not able to pass along the accumulated knowledge. It may be that so many of the older papers and research and real-world experience are not available online and, hence, not findable with a quick Google search. Or it may be that the computer science and engineering curricula aren't covering the history of computing at all, let alone the history of secure computing. Whatever the reasons, we're losing a lot of time and energy rediscovering technology and re-visiting the same failed solutions over and over again.

“it depends”:

defining legal values for network behavior

by Erin Kenneally

Erin Kenneally is a Forensic Analyst with the Pacific Institute for Computer Security (PICS), San Diego Supercomputer Center. She is a licensed attorney who holds Juris Doctorate and Master of Forensic Sciences degrees.

erin@sdsc.edu

> “Men feed data to a computer and men interpret the answer the computer spews forth. In this computerized age, the law must require that men in the use of computerized data regard those with whom they are dealing as more important than a perforation on a card. Trust in the infallibility of a computer is hardly a defense, when the opportunity to avoid the error is as apparent and repeated as was here present.”

This was a response by a Court of Appeals to Ford Motor’s assertion that a computer mistake caused it to wrongfully repossess a customer’s car.[1] Although the punch card reference might cause one to dismiss its significance as antiquated pre–World Wide Web naïveté, the underlying message remains even more true in our current Internetworked society. Whether we are dealing with a disputed bill payment and mistaken repossession or questionable computer access and release of protected information, human use of computers involves conflicts over values and property that necessitate defining and enforcing socially acceptable behavior. This is where the standard known as “reasonableness” is paramount to guiding the acts and consequences involved in network behavior.

Ignorance Is Not Bliss

On one hand, people are quick to parrot the notion that Internetworked society is the new Wild Wild West, referring to the dearth of computer-specific laws and regulations, coupled with our free-market tradition of allowing industry to self-regulate and reward beneficial behavior. Nonetheless, even the habitation of “unchartered” land relied upon notions of reasonableness to guide the resolution of conflicts. To continue to use the relative dearth of historical network behavior as a justification for unacceptable behavior is to ignore the social indicators of consensus on what network behavior is tolerable.

“Reasonableness” itself is a relative and dynamic standard in that there is no fixed formula, and determinations are made on a case-by-case basis.[2] Yet this standard underlies many of our laws and constitutional rights. Reasonableness can be found at the heart of many mechanisms that are invoked to govern network behavior – laws such as the Computer Fraud and Abuse Act (CFAA), the Fourth Amendment, social pressures (i.e., public relations), and corporate privacy policies.

Recognizing that there are myriad standards regarding computer security and yet no single, overriding measuring stick, one is hard-pressed to know what standard should be followed so as to not run afoul of the law. Furthermore, the “reasonableness” standard facilitates the ignorance defenses, blaming the victim, and Robin Hood justifications, as evidenced by the cases discussed below.

Nevertheless, both the physical and digital realms employ laws, contracts and licenses, and informal social pressure as mechanisms to notify and implement acceptable behavior. However, a notion of what constitutes acceptable computer network behavior is much less developed. For example, there is no disputing that taking a golf club to my neighbor’s window is unacceptable; yet employing an equally malicious software tool against a fellow Netizen’s computer does not necessarily evoke such a binary judgment of right and wrong. Although consensus about acceptable network behavior is

1. *Ford Motor Credit Company v. Swarens*, 447 S.W.2d 53 (1969).

2. What is a reasonable search is not to be determined by any fixed formula. The Constitution does not define what are “unreasonable” searches and, regrettably, in our discipline we have no ready litmus test. The recurring questions of the reasonableness of searches must find resolution in the facts and circumstances of each case. *United States v. Rabinowitz*, 339 U.S. 56 at 63 (1950).

embryonic and there are comparably fewer laws that are specific to "cyber-behavior," notions of right and wrong do exist. Recent conflicts involving Ziff-Davis, HP and Snosoft, Princeton and Yale universities, and ForensicTec illustrate how we are defining reasonableness in our Internetworked society.

Law as a Metric for Reasonableness

If reasonableness is a consensus standard, what is the relevant metric? Although bad PR/public opinion is not a formal category that courts check off when adjudging reasonableness, it nonetheless can be a gauge of what society believes to be right and wrong behavior. Other metrics for ascertaining reasonableness include regulations, policies and practices, contemporary litigation, notice/knowledge, and capability.

As with ForensicTec, employees accessed government and other private networks and viewed and downloaded files containing military procedures, email, SSNs, and financial data.[3] Attempting to justify their actions, ForensicTec allegedly noted that they used publicly available scanning software to identify vulnerable computers, as well as using easily guessed passwords to gain "unobstructed" access to these sites.

In response to criticism and allegations that it had violated the federal computer crime law (CFAA), ForensicTec played the Robin Hood card by claiming that it was merely pointing out serious vulnerabilities in various networks. Even though ForensicTec was a "legitimate" company on paper, its actions were no different from that of a teenaged hacker conducting the digital equivalent of chest beating. In the eyes of the law, the same elements that constitute a crime are present in both cases: intentional access to a protected computer without authorization.[4] As for the intent element, ForensicTec admitted to repeatedly navigating through multiple government and private networks, thus illustrating knowledge and directed control of its scanning, cracking, and probing activities for connecting to and entering other systems. Insofar as "protected" has been interpreted to mean any computer connected to the Internet, that element can be checked off. And finally, unless ForensicTec had some type of agreement or consent from the government agencies and companies that it accessed, its digital exploration was unauthorized.

Never mind that it notified the vulnerable victims *after* contacting the media, ForensicTec had no legal right to troll through networks where they had no legitimate and authorized business reason to be. Is this any different from trying to justify strolling through your neighbor's house because you discovered their door was open? The ease with which one can access another's property, be it their computer or homestead, is not a factor in determining whether one's actions are lawful. The situation would be no different if ForensicTec had used a million-dollar, one-of-a-kind software tool and/or the victim computers were locked 50 feet below the Pentagon behind a 24/7-managed intrusion detection-firewall schema. Furthermore, although motivation may be a mitigating factor in the penalty phase, the law only considers the "why" insofar as it can be used to infer proof of some element (act or intent) of the offense. This is why Robin Hood and Jean Valjean were both criminals, despite the honorable motivations behind their acts.

Defining Expected Network Behavior

Another emperor without clothes was sighted in the case of the Princeton University administrator who used the SSN and birth dates of student applicants to access admission records at Yale University.[5] Similar to the ForensicTec case, Princeton's actions

3. See Robert O'Harrow, "U.S. Probes Firm in Security Breach: Consultants Invaded Federal Computers," Washington Post.com (August 21, 2002): *http://www.washingtonpost.com/wp-dyn/articles/A42019-2002Aug20.html*.

4. See 18 U.S.C. § 1030.

5. See Karen Arenson, "Princeton Pries into Web Site for Yale Applicants," New York Times Online (July 26, 2002): *http://www.nytimes.com/2002/07/26/education/26IVY.html*.

invoke laws that decry and punish Princeton's network behavior. Primarily, the federal Computer Fraud and Abuse Act prohibited Princeton's intentional entry into Yale's Web site without authorization.[6] The official in question admitted to using the students' identification information to call up their application status from the non-public, restricted Web site, which provided notice that it was authorized for use by prospective students only.

Notwithstanding a seemingly clear violation of the law, the public outcry in response to Princeton's actions is perhaps a stronger metric for the unreasonableness of its network behavior. Princeton officials were quick to express regret and exact discipline almost as soon as the news hit the academic community. In this respect, there was no disagreement that Princeton's digital behavior was illegal and unethical. However, a great deal of attention was shifted by legal scholars to what this situation said about the state of competitiveness in Ivy League admissions. Surely in this breach of student trust the very act of digitally trespassing onto the property and values of another should be the primary focus, rather than the perceived symptom of collegial competitiveness.

The medium of storage or method of transmission should not alter one's (lack of) right to interfere with another's property or values. Would we tolerate a Princeton official using the same information to request and obtain physical records contained in a file cabinet or to greet the postal carrier at the students' mailboxes under such false pretenses? If we bear in mind this non-distinction between traditional and cyber-actions when assessing acceptable network behavior, it is harder to get sidetracked by defenses that obscure the wrongfulness of the act. To do otherwise would dismiss unauthorized intrusive acts as "gaining access out of curiosity about a site's security" or lead down the slippery path of blaming the victim. To be sure, Yale could have implemented stronger security measures such as PINs or randomly generated identifiers to secure applicant records. Other laws such as the Federal Education Records and Privacy Act would give applicants a reasonable expectation that their personal records would not be misused or unprotected by the schools. However, Yale's duty to protect its students' information is an issue distinct from whether Princeton acted with knowledge that it was not permitted to digitally trample on Yale's property rights and values. If someone enters your house and snatches your child's savings bonds – regardless of whether he climbs through an open window or blasts a battering ram through the steel-fortified door – he is violating your reasonable expectations to be secure in your home, as defined under the burglary laws (your child's ability to afford college being a separate matter).

Policy: Another Metric for Reasonableness

Laws and regulations present relatively clear metrics for standards of tolerable behavior, since they are supposed to be formal embodiments of society's values. Private policies, licenses, and contracts are other mechanisms by which society defines, enforces, and adjudges reasonable network behavior. Insofar as individuals can more easily dictate the scope of policies and contracts, these mechanisms may more accurately reflect and more immediately shape notions of reasonableness.

The expectation of online personal data privacy, defined by Web site policies and enforced by informal consumer sanctions, is a prominent instance of acceptable network behavior. This was illustrated in the case involving the Ziff-Davis settlement payment of $150,000 to various states and customers for exposing the credit card

6. In addition, Princeton may have run afoul of the Federal Education Records and Privacy Act (FERPA; 20 U.S.C. 1232g (1993), regulations at 34 C.F.R. 99 (1993)), which creates minimum standards for educational institutions receiving federal funds to protect students' records. FERPA considers student SSNs an education record in and of itself. SSN collection and disclosure by government agencies is generally prohibited by the Privacy Act of 1974.

numbers and identifying information of many of its subscribers.[7] Here, expectations of reasonable behavior were primarily derived from Ziff-Davis's privacy policy, which promised reasonable security in protecting its customers' financial and identity data stored in its databases.[8]

Policies that speak to protecting customer private financial information create a bilateral notice that one party is agreeing to disclose data to the other party that is not for public consumption. Alongside this knowledge is the expectation that the party receiving the data has the capability to protect it from prying eyes, as well as the duty to enforce the promise. These expectations are not unique to Internetworked society – they speak to the same "reasonableness" that demands notice, consent, security, and enforcement of rights to control physical property. If you give your Visa card to the clerk at Krispy Kreme, you reasonably expect that the information will not be posted to the telephone pole outside. In fact, you feel entitled to have that information protected by the physical store owner.

Likewise, the level of computer literacy in the public is such that people have transferred that sense of entitlement to the transmission of private information over the Internet. As such, vendors are expected to implement technical measures to ensure that this information is not exposed to the public. Increased publicity surrounding identity theft has certainly added force to that reasonable expectation.

In fact, the Federal Trade Commission's regulation of unfair and deceptive trade practices is heavily influenced by expectations established by privacy policies.[9]

The expectations become less clear regarding the extent to which a company posting a privacy policy must go to protect the customer data. This will likely be resolved on a case-by-case basis, depending on how specifically the policy was worded, as well as what the cost-benefit analysis reveals. In the Ziff-Davis case, the data was readily exposed by anyone engaging in normal Internet surfing, as opposed to having been unlawfully accessed by someone with über-hacker skills. Despite the fact that Ziff-Davis claimed this was a result of a coding error, it's likely that the potential negative publicity – with corresponding loss of good will and customer base – factored into its decision to pay for its insecure posture. Thus, the risk of damaging commercial reputation and profitability are ways that informal social pressures can be a metric for defining reasonable network behavior.

Reasonableness as a Balancing Act

Amid the cases discussed thus far, a consensus of "reasonable" network behavior has been relatively clear because analogies could be drawn from notions of right and wrong surrounding physical property. The Snosoft-HP case involves conflicting standards of reasonableness, emanating from different measuring sticks. The issue in this case was the reasonableness of disclosing a computer security vulnerability. Snosoft researchers uncovered a vulnerability in the Tru64 UNIX operating system distributed by Hewlett-Packard (HP).[10] Before disclosing the exploit to the public, Snosoft followed the informal custom of informing HP so as to give it time to develop a patch and disseminate it to the community of users. As a result of HP's failure to respond, a Snosoft researcher alerted the public to the flaw. In response, HP threatened Snosoft with violations of the federal Digital Millennium Copyright Act (DMCA) and Computer Fraud and Abuse Act.

On one hand, Snosoft claimed its actions were a reasonable application of its fair use rights under copyright law, the First Amendment right to free speech, and a reasonable

7. See, generally, Seanna Adcox, "Ziff Davis Agrees to Pay Settlement," Findlaw News and Commentary (visited August 29, 2002): *http://news.findlaw.com/ap_stories/f/1310/8-28-2002/20020828141503_76.html.*

8. See *http://www.ziffdavis.com/terms/index.asp?page=privacypolicy*.

9. Federal Trade Commission, "Privacy Online: A Report to Congress" (June 1998): *http://www.ftc.gov/reports/privacy3/priv-23a.pdf.*

10. See Declan McCullach, "Security Warning Draws DMCA Threat," CNET News.com (July 30, 2002): *http://news.com.com/2100-1023-947325.html.*

means to protect themselves and other Tru64 UNIX community members. HP invoked other legal mechanisms to justify the reasonableness of its actions to protect its copyrighted property. Although the debate surrounding the DMCA is well-published, contentious, and beyond the scope of this article, the significance of these conflicting values is that informal social pressure prevailed over institutionalized legal standards in defining reasonable network behavior.

Despite having the force of law and case precedent (to date, every lawsuit challenging the DMCA has failed) on its side regarding the DMCA's broad prohibition on circumventing copyright protection technologies, HP backed off of federal charges against the researchers who published the exploit. The case for Snosoft's reasonableness boiled down to the public support for this socially beneficial act of alerting potentially harmed parties who were denied knowledge and protection from the vendor. The power of this informal social pressure was surely manifest in the remnants of bad publicity that befell Adobe when it was embroiled with Dmitry Sklyarov over a similar issue. Similar to Ziff-Davis, the social sanctions (harm to reputation and commercial profitability) turned out to be a formidable ally in reaching some sort of consensus on right and wrong Internetwork behavior. HP's prior knowledge of the flaw, capability to rectify the vulnerability, wait-and-see approach, and reactionary attempt to use the law to counterattack did not gain it social popularity points.

Just as the Fourth Amendment has become the de facto reference point for defining "reasonableness" (in the search and seizure context) and has been interpreted by the US Supreme Court as a balancing test between conflicting interests, a similar weighing is evolving in the context of network behavior.

This is undoubtedly not the last time we will encounter the conflict between rights and property that DMCA brings about. Perhaps the introductory case of *Ford v. Swarens* holds greater significance and foresight than we are willing to acknowledge. As that scenario played out, the collectors returned for the last time to collect from Swarens, who, Ford admitted, was always current on his payments. Frustrated with their groundless visits, Swarens advised them that he would show them no more records, and strongly suggested that they leave his home . . . while brandishing a shotgun. They left, but first reminded him of their experience in the repossession of automobiles and promised him that they would repossess his.[11]

As we know, however, the court had little sympathy for the computer-error defense raised by Ford. Instead, it chose to use the context-dependent standard of reasonableness – a standard that humans must continually define if we wish to resolve digital repossessions and cyber gun-slinging conflicts that are sure to arise in our increasingly technology-centric interactions with one another.

11. *Ford Motor Credit Company v. Swarens*, 447 S.W.2d 53 (1969).

active network defense

by Sven Dietrich

Sven Dietrich is a Member of the Technical Staff at the Carnegie Mellon Software Engineering Institute in Pittsburgh, PA. He is the project manager for the Active Network Defense project, and focuses his research on security and survivability.

spock@cert.org

Some Concepts and Techniques

We define Active Network Defense (AND) as the defensive side of computer network operations. The defensive side contains active and passive aspects, and we will focus on the active techniques available for defensively engaging the attacker.

The threats that AND is attempting to address include:

- Denial-of-service (DoS) attacks, including distributed DoS attacks, which cause unusable or crashed systems, clogged networks, or untimely responses for critical missions.
- Worms, recognized as a threat to network security when the Morris worm spread in 1988. Worms such as Code Red and Nimda continue to pose a real threat through their propagation and their embedded mission.
- Viruses which attach themselves to electronic mail messages and local documents.
- Malicious code, including various attack techniques not covered above, such as penetration via exploits, external information gathering such as scanning, etc.

In order to address the threats, one must consider different approaches. The goals set forth are to limit access for the attacker by reducing the attacker's ability to do damage by changing:

- the network path from the attacker to the victim
- what the attacker can see and/or do.

Active Deception Techniques

Typical deceptions include concealment of information, false and planted information, and camouflage to mislead the attacker with respect to characteristics or contents of a host or network. The attacker's strength is focused on an interesting object, such as a host, a set of hosts, or an entire network of attractive hosts. Fred Cohen's Deception ToolKit[1] is a prime example of this honeypot capability. More recently, highly interacting networks of honeypots have been used in the Honeynet Project[2] to discover presence, tools, tactics, and the intent of the attacker. By actively feeding the attacker more attractive systems in comparison to the systems to be defended, the attacker is encouraged to spend time and energy compromising, exploiting, and contacting these honeypots. Since honeypots are not production machines, should a connection from a honeypot be detected it would indicate the presence of the attacker. A recent development is the concept of a virtual honeypot (honeyd).[3] In this approach, a single host creates virtual hosts on a network, with given characteristics of a chosen operating system, in order to deceive the attacker, including a given topology of virtual hosts. Here is a sample configuration script for honeyd:

```
annotate "AIX 4.0 - 4.2" fragment old
# Example of a simple host template and its binding
create template
set template personality "AIX 4.0 - 4.2"
add template tcp port 80 "sh scripts/web.sh"
add template tcp port 22 "sh scripts/test.sh $ipsrc $dport"
add template tcp port 23 proxy 10.23.1.2:23
set template default tcp action reset

bind 10.21.19.102 template
```

1. Fred Cohen, Deception ToolKit: *http://www.all.net/dtk/index.html*.

2. The Honeynet Project: *http://project.honeynet.org/*.

3. Niels Provos et al., virtual honeypots: *http://www.citi.umich.edu/u/provos//honeyd/*.

The config script simulates an AIX host with the old fragment reassembly policy (to fool scanning tools such as nmap),[4] scripts to handle probes/connects to ports 80 (HTTP) and 22 (SSH), and the capability to proxy port 23 (telnet) connections to another host. Any other TCP scanning attempts will encounter a TCP reset. Even though honeyd is a work-in-progress, it can be downloaded and is usable today.

Related to the idea of obfuscation outlined before, there is another deception technique, known as packet scrubbing,[5] which can hide and falsify host, operating system, or other characteristic information for those systems behind the packet scrubber. An attacker looking to fingerprint a target host or network will be faced with false and possibly inconsistent information. Consider a network of mixed hosts, say PCs and Macintoshes. One can make the network look like all PCs, all Macintoshes, all Solaris boxes, or like nothing in particular. The emphasis, however, is on "normalizing" the traffic so that it is indistinguishable from any other operating system, rather than pretending to be a different type of operating system. One approach is already built into the pf packet filter,[6] included in OpenBSD, for example. A simple configuration line added to your pf.conf file will "scrub" your inbound traffic in an effort to thwart exploitation of ambiguities in TCP/IP protocol stacks to perform fingerprinting attacks or worse:

```
ext_if = "kue0"

# normalize all incoming traffic
scrub in on $ext_if all
```

Attacks that are worse include exploiting the IP fragment reassembly techniques of intrusion detection systems for overlapping IP fragments. Using pf with packet scrubbing enabled on a NAT (Network Address Translation) box or firewall will protect hosts on the closed side. OpenBSD itself is already immune to such attacks.

Hogwash takes a more proactive approach. Using snort-like configuration files, Hogwash is built on top of layer 2, also known as the data link layer, and is designed to run on Linux systems without IP networking installed, so as to be completely invisible on the network. The defense philosophy of Hogwash is centered on the theory that a low-level network approach will prevent the packet scrubber from becoming the target of the next attack. Its focus is to drop or sanitize malicious packets only. All other packets travel completely unmodified through the network, since the system does not directly interact with the packet at the protocol level (e.g., Ethernet hardware addresses or time-to-live fields do not get changed or updated). Packets fall into three categories:

- Legitimate or good packets are let into the network.
- Malicious or bad packets are dropped by the system, and an alert is sent to the operator.
- Transient or, sometimes, bad packets are left unaffected, but an alert is still sent to the operator.

For example:

```
drop tcp $EXTERNAL_NET any -> $HOME_NET 80 (content:"/etc/passwd";
msg:"WEB: attemp to request /etc/passwd";)
```

This drops any requests originating on the external network and directed at the Web server on the home network that contain the string /etc/passwd, potentially an attempt to retrieve the UNIX password file. The Hogwash project is still experimental, but its author claims that a Celeron 733-equipped host with two 100Mbps Ethernet

4. Fyodor, nmap: *http://www.insecure.org/nmap/index.html*

5. Matthew Smart, Robert Malan, and Farnam Jahanian, "Defeating TCP/IP Stack Fingerprinting," *9th USENIX Security Symposium*, 2000: *http://www.usenix.org/publications/library/proceedings/sec2000/smart.html*.

6. Daniel Hartmeier, "Design and Performance of the OpenBSD Stateful Packet Filter (pf)," *2002 USENIX Technical Conference*, June 2002: *http://www.benzedrine.cx/pf.html*.

cards can handle the full 100Mbps network, depending on the rule set. Your mileage will vary.

Preemptive Strike

As a preventive measure, network defenders can actively wander the networks in search of potential attackers. Once a potential attacker has been identified, by whatever means, the network defenders can collect intelligence on the capabilities of the attackers. Such intelligence gathering can include host and network characteristics – for example, operating system versions, infrastructure architecture details, and router operating systems. By exploiting this knowledge, the defenders could make a preemptive strike against the attacker, taking advantage of existing vulnerabilities in the remote hosts. This can take several forms. If a potential set of vulnerable hosts is identified and it is known they are not (yet) under the control of the attacker, then a series of "mass patchings" can remove vulnerabilities in the remote machines. If, on the contrary, the hosts have already been compromised, then some cleaning code can be injected to remove the malicious code and possibly patch the system. This was demonstrated in the response to Code Red: a benign version of the original Code Red called Code Green[7] removed Code Red from an attacking system and patched it to resist further Code Red infections. The development of Code Green was assisted by the findings[8] of the Eeye Digital Security team, and by various other "experiments" with the Code Red worm. It is, of course, possible to turn the hostile host against its controller, effectively turning the attacker's agents against the attacker's host(s).

Rather than attacking the enemy's hosts themselves, it is also possible to target the routing infrastructure, resulting in a collapse of the attacker's connectivity. Cooperative ISPs can install access control lists and/or rate-limiting to prevent attack traffic from entering the Internet core. Lacking ISP cooperation, DDoS or crafted BGP attacks against routers can cripple the connectivity of attacking hosts. BGP has been the object of analysis for its instability,[9] but seriously: "Kids, don't try this at home."

Striking Back at the Attacker

In a different scenario, such as during an ongoing attack, network defenders could explore the possibility of retaliating against the attacking hosts. While this is a legally and ethically problematic subject,[10] let us explore what the possibilities are today.

The first option would be to disable the attacking machines, if/when they have been properly identified. Suppose such traceback or other identification has taken place, then fault inducement in the attacker's code, the underlying environment, and/or the operating system will stop the attack, or at least a portion thereof. By exploiting known vulnerabilities causing kernel panics in the remote host, such as the Ping of Death[11] or teardrop,[12] the attack would stop, temporarily at least. By gaining system privileges on the remote host, one can modify, clean, or spoil the system for the attacker for extended periods of time.

The second option would be to attack the immediate surroundings of the attacker. If for some reason the attacking hosts are impenetrable, disabling the nodes providing connectivity to the attacking hosts would make the attack stop. This can be achieved by crashing routers or causing the local routing infrastructure to collapse.

A third option would be to develop code that leverages strategic knowledge of the attacker's intentions and techniques to thwart the attack. Again using the Code Red incident as an example, the re-addressing of the whitehouse.gov servers which allowed

7. Herbert HexXer, Code Green: a copy is available at *http://archives.neohapsis.com/archives/vuln-dev/2001-q3/0575.html*.

8. Eeye Digital Security. ".ida 'Code Red' Worm": *http://eeye.com/html/Research/Advisories/AL20010717.html*.

9. Rik Farrow, "Routing Instability on the Internet," Network Magazine, March 2002: *http://www.networkmagazine.com/article/NMG20020304S0007/2*.

10. Nathan Buchheit, Anthony Ruocco, and Donald Welch, "Strike Back: Offensive Actions in Information Warfare," New Security Paradigms Workshop, 1999.

11. CERT, Advisory CA 1996-26: *http://www.cert.org/advisories/CA-1996-26.html*.

12. Microsoft Corporation, "Stop 0a in tcpip.sys When Receiving Out-of-Band (OOB) Data": *http://support.microsoft.com/support/kb/articles/Q143/4/78.asp*.

the network defenders to sidestep Code Red's date-triggered DDoS attack and the development of Code Green, were made possible by the analysis and scrutiny of Code Red's behavior produced by many cooperating analysts and reverse engineers. The recognition of the attacker's mission enabled response options otherwise not considered.

Dynamic Infrastructure Modification and Traceback

The aftermath of both the University of Minnesota (1999) and the February 2000 attacks generated a series of DDoS traceback and mitigation schemes. The following is a list of a few representative examples based on their predominant role, end host vs. infrastructure.

Schemes Within the End Hosts

Both Bellovin[13] and Savage et al.[14] show how information fed back to the attacked hosts facilitates attack path reconstruction. Since the audit messages are sent probabilistically (typically one every 20,000 packets), the victim, having accumulated enough of these, can trace the real path back to the attacker, who is attempting to evade tracking by spoofing its source address. Song and Perrig[15] improve on Savage's work, providing more efficiency and scalability. Both schemes incorporate messages into unused fields of the IP packets, effectively marking them. All of the schemes involving the marking of small numbers of packets are problematic for a very widespread attack using small numbers of packets from a very large number of machines.

Snoeren et al.[16] propose a passive monitoring scheme for assisting with the traceback of a single packet. Packet digests are kept for a limited amount of time and permit the traceback across the *traceable* infrastructure up to the edge of this tracing infrastructure. The scheme provides a low false positive rate, which diminishes as the packet moves closer to the source of the offending packet. The impact is minimal, since the mechanism attaches to the network passively, but it can be implemented on the router. Note that the schemes of Bellovin and Snoeren require that additional traffic be delivered to the host under attack in order to determine the attacking location. This is problematic under conditions where links or routers are saturated, but it is conceivable to perform this via an out-of-band mechanism.

Schemes Within the Infrastructure

All the schemes in this category suffer from various degrees of the same shortcoming: a substantial latency involved in recognizing an attack and implementing a countermeasure. Short-lived, or one-packet, attacks cannot easily be handled with these techniques. In addition, it may be the case that attacks that occur in bursts with durations shorter than the countermeasures' recognition and reconfiguration period will largely evade the countermeasures. Similarly, it is not clear that the approaches are viable if an attack comes from a very large number of well-dispersed sources. Note that any of the traceback techniques of the previous section could be used to guide these countermeasures. Stone's CenterTrack[17] uses an overlay network of routers that allows for monitoring and rerouting of suspicious traffic. Bellovin and Ioannidis implement pushback,[18] a router-based mechanism that treats DDoS as a congestion-control problem and drops the traffic causing the congestion. Sterne et al. propose an active network approach in their autonomic response to DDoS attacks.[19] Malicious attacks are countered by sending mobile code upstream, which analyzes traffic flows on each router and duplicates itself at split points until the source of the offending stream is narrowed down or identified. Papadopoulos et al. investigate the coordinated

13. Steve Bellovin, Marcus Leech, and Tom Taylor, "ICMP Traceback Messages," IETF work-in-progress, October 2001.

14. Stefan Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical Network Support for IP Traceback," in ACM SIGCOMM 2000.

15. Dawn Song and Adrian Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," IEEE Infocom 2001.

16. Alex Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer, "Single-Packet IP Traceback," *IEEE/ACM Transactions on Networking* 2002, forthcoming.

17. Robert Stone, "CenterTrack: An IP Overlay Network for Tracking DoS Floods," 9th USENIX Security Symposium, 2000: *http://www.usenix.org/publications/library/proceedings/sec2000/stone.html*.

18. J. Ioannidis and Steve Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," Network and Distributed Systems Security Symposium, February 2002.

19. Dan Sterne, Kelly Djahandari, Ravindra Balupari, William La Cholter, Bill Babson, Brett Wilson, Priya Narasimhan, Andrew Purtell, Dan Schnackenberg, and Scott Linden, "Active Network-Based DDoS Defense," 2002 DARPA Active Networks Conference and Exposition (DANCE 2002), May 29–31, 2002.

approach to dealing with DDoS in their COSSACK scheme.[20] The correlation between the attacking hosts, i.e., the simultaneous presence of similar packets, reveals the presence of an attack in this snort-based tool. D-WARD[21] looks at the validity of TCP connections, such as completed three-way handshakes, for allowing or disallowing packets through routers. By establishing traffic models, potentially offensive packets are kept at bay via throttling at the router level.

Most of this work assumes that the attacks and their sources have been correctly identified before performing what amounts to a denial of service on itself. The risk remains that due to a partial compromise of the system a denial of service is easily triggered, effectively finishing the task intended by the attacker.

On a more practical note, UUNET has developed an interesting technique for dealing with traceback of an attack flood, called "backscatter traceback."[22] In this technique, a clever combination of BGP configuration and triggers can quickly lead to the entry point into the infrastructure of spoofed IP addresses. During an attack, the offensive traffic is redirected to a null interface on the border routers. The resulting flurry of ICMP unreachable messages is sent back to both legitimate and spoofed sources, and a large portion of these messages destined for non-routable addresses (a large chunk, say 96.0.0.0/3) are redirected to a so-called sink hole network. Since the source address of these ICMP messages is one or more routers, which represent the entry points into the infrastructure, the source of the attack can easily be traced and quenched, often within one or two minutes.

A Comment in Closing

While some of the techniques in AND (the preferred term is now Computer Network Defense Response Actions or CND RA) remain controversial, it provides fertile ground for research as countermeasures are challenged and circumvented in a continuous cat-and-mouse game.

ACKNOWLEDGEMENTS

I would like to thank John McHugh, Howard Lipson, Eric Hayes, and other colleagues at CERT for their contributions and comments, and Tom Longstaff for making this article possible.

20. Christos Papadopoulos, Ramesh Govindan, Bob Lindell, and John Mehringer, "Coordinated Suppression of Simultaneous Attacks (COSSACK)," December 2001: *http://www.isi.edu/cossack/*.

21. Jelena Mirkovic, Peter Reiher, and Gregory Prier, "A Source Router Approach to DDoS Defense," Technical Report 010042, UCLA Computer Science Department, 2001.

22. Brian W. Gemberling, Christopher L. Morrow, and Barry R. Greene, "ISP Security: Real-World Techniques," October 2001: *http://www.nanog.org/mth-0110/greene.html*.

the regional information watch

by Abe Singer

Abe Singer is a computer security manager at the San Diego Supercomputer Center, and occasional consultant and expert witness. His current work is in security measurement and security "for the life of the Republic."

Abe@SDSC.edu

Introduction

The San Diego Regional Information Watch (SDRIW)[1] is a sort of "network neighborhood block watch," a venue for bringing together area system/network security people and law enforcement, providing education about the technical and legal issues surrounding computer security, and providing an opportunity for "human networking." It has been very successful, and with the ever growing need for effective security and investigation of intrusions, we would like to share our experience and encourage others to form their own regional information watch.

Evolution

SDRIW began in part as a reaction to the incident of Steve Jackson Games and Operation Sundevil.[2] University system administrators (and others) were hesitant to report intrusions to law enforcement for fear that their systems would be seized for long periods of time. We were looking for a way to get these groups talking when an opportunity appeared: the mayor's "City of the Future" program[3] realized that there was a need for some type of high-tech law enforcement to be ready to protect the city's new high-tech sector.

SDRIW was originally thought of as a regional CERT;[4] however, we discovered that providing introductions and building trust between law enforcement, academics, and companies was more important and beneficial than being a CERT would have been.

SDRIW started as a three-way partnership between the San Diego Supercomputer Center (SDSC), Space and Naval Warfare Systems Command (SPAWAR),[5] and San Diego Data Processing Corporation (SDDPC).[6] SDDPC has faded from the scene, but we still have ties to SPAWAR, and our participation from law enforcement and local high-tech companies has grown. In the last couple of years, our meeting attendance has averaged around 40 people. We have over 200 addresses on our mailing list. Our "members" include system and network administrators, consultants, federal and local law enforcement, attorneys, students, researchers, local press, and even members of the local 2600 chapter.[7]

And we have realized our goal. Members are familiar with each other. If I have an incident at 3 a.m. and think it warrants attention from the FBI, I can get an agent to return my call. (We've actually had to do this.) Likewise, we've been called by some of our friends at other companies when they were in the thick of it. People are actually talking to and helping each other.

Mission

The Regional Information Watch is a combination of network neighborhood block watch, users group, and information-sharing organization.

We provide opportunity for:

- Information exchange/information sharing on technical and legal issues relevant to computer security.
- Face-time and familiarity between area professionals and law enforcement.
- "Networking" opportunities for people in the computer security arena.
- Early warning of regional incidents.
- Education on security tools, techniques, and standards.

The Regional Information Watch is open to anyone who wishes to participate.

1. San Diego Regional Information Watch: *http://www.sdriw.org.*

2. Bruce Sterling, *The Hacker Crackdown*: *http://www.mit.edu/hacker/part2.html.*

3. "San Diego: City of the Future — The Role of Telecommunications," Report of the Mayor's Advisory Committee on the City of the Future, San Diego, March 11, 1994: *http://www.smartcommunities.org/city_of_the_future_report.pdf.*

4. Computer Emergency Response Team: *http://www.cert.org.*

5. Space and Naval Warfare Systems Command: *http://enterprise.spawar.navy.mil/spawarpublicsite/.*

6. San Diego Data Processing Corporation: *http://www.sddpc.org/.*

7. San Diego 2600: *http://www.sd2600.org/.*

Activities

Currently, the group's only activity is to hold monthly meetings, and we maintain a Web site with some local information, such as contacts in the area.

At our meetings we have had presentations on software security tools, the USA PATRIOT Act, forensics, California computer crime laws, data recovery, and handling email and phone threats, just to name a few. Some of our presentations are quite technical, others are for beginners.

But we don't just do presentations; in fact, we usually have 20 to 30 minutes of discussion before the "feature presentation." Our meeting agenda looks something like this:

- Introductions
- Announcements
- Current events
- Incidents, vulnerabilities, exploits
- New security tools
- Upcoming conferences and other events
- Featured speaker – varying topics, varying levels
- Call for speakers
- "Social hour"(sometimes with food)

The meeting is all in an open, round-the-room format, which gives people the opportunity to ask, "Has anyone ever seen probes like X?" or "Where can I find a tool to do Y?" or "We have a job opening for Z."

Our law enforcement members actively participate, letting us know about things happening on the legal side, and sometimes telling us about interesting cases that have been closed.

While our presentations are always useful, the social hour is often especially so, providing an opportunity for people to chat and get to know each other (some like to call this "networking").

And we occasionally act as a point of contact for related issues; we will get a call from law enforcement or a local company asking, "Do you know whom to contact at company X?"

Members/Structure

Officially, SDRIW doesn't exist. We're just a "public meeting," a group of people peaceably assembling to chat about interesting stuff. There is no board, no corporate entity – just some people putting stuff on a Web site and mailing out information about who might be somewhere at a certain time. We chose that path on purpose; there's no one to point a finger at, no membership requirements, restrictions, or dues (for that matter, no official membership).

Our "membership" consists in part of:

- Academics: sysadmins, researchers, and students from UC San Diego and San Diego State University.
- Large companies in the area: Qualcomm, SAIC, Cox Communications, TRW, SPAWAR, Exodus.
- Small companies: Anonymizer.com, local ISPs, independent consultants, forensics companies.

- Hardware vendors: Sun and others.
- Law enforcement and government: FBI, Secret Service, Postal Service, US Attorney's Office, City Police, County Sheriff, University Police, County D.A.
- Community: local members of 2600.

Future Goals

While we feel we have accomplished our primary objective, we do have some goals for expanding our activities. We are looking for sources of funding to allow us to pay expenses for out-of-town speakers and eventually host some workshops or a conference.

We would also like to provide job listings for security-related positions in the area.

And, of course, we always want to expand our "membership." We'd like to get more participation from ISPs and other companies in the area. In an ideal world, we'd have a security contact for every ISP in town, and for the security or sysadmins for (at least) the larger networks in the area.

Competition

When telling people about SDRIW, I am sometimes asked, "Isn't that what HTCIA (High-Technology Crime Investigation Association)[8] does?" Well, yes and no.

San Diego does have an HTCIA chapter. There is also a San Diego chapter of Infragard,[9] and a chapter of ISSA (Information Systems Security Association).[10] Many of the SDRIW participants are members of these groups too. In other cities, HTCIA or Infragard chapters might provide some of the services that SDRIW provides. But in San Diego, SDRIW was already so successful that the local Infragard chapter decided that it would not try and reproduce the effort in the same area; instead, it suggests that Infragard members interested in computer security participate in SDRIW.

HTCIA is roughly 80% law enforcement and 20% technical people. HTCIA is a formal organization with annual membership dues. New members have to be endorsed by existing members and consent to a background check. HTCIA's scope is wider than just computer security and is focused on criminal/legal aspects.

Infragard is also wider in scope than just computer security – its members are involved with "critical infrastructure" organizations. The San Diego chapter does not require dues but, like HTCIA, does background checks and has confidentiality requirements.

ISSA requires dues ($100 in San Diego).

SDRIW is about 80% technical people, 20% law enforcement, is focused on computer security, and is free.

How to Form Your Own

Okay, so you're sold, and you'd like to form your own group. Here's how to do it:

Find a space for a meeting. Maybe your employer has an auditorium or a large meeting room; someplace with a video projector for laptops is best; Internet access is helpful.

Pick a time for your first meeting. You'll eventually want to schedule a regular time, but start with a single meeting. We schedule our meetings from 2 to 4 p.m., which gives some people the opportunity to avoid having to go back to the office afterwards.

8. High Technology Crime Investigation Association: *http://www.htcia.org.*

9. Infragard: *http://www.infragard.net/.*

10. Information Systems Security Association: *http://www.issa.org.*

If you can, find someone to give a presentation on something interesting. Do one yourself if you can't find anyone else. Or come up with a discussion topic for the group.

Put up a Web site (it can be simple to begin with) and mailing list for announcing meetings. Our mailing list is moderated and is used almost exclusively for the meeting announcements, so volume is quite low. We have found that some people rely on the mailing list to know when the next meeting is, and some look at the Web site. So it's useful to have both. Put information about the meeting on your Web site, including presentation topic and directions to the meeting. Also include information on subscribing to the mailing list.

Invite people to attend. For people you know, you might be able to get away with an email. But for people you don't know, you might want to pick up the phone and talk to them. They'll take you more seriously. Especially law enforcement people.

Who should you invite? Well, start with local companies. Call all the ISPs in town and ask to talk to their security person. Find security consulting firms and invite them. If you've got any colleges or universities in town, find a contact there, and ask them to forward it on to others. Identify companies with large networks and contact their security or network administrators, and/or CIOs.

Go visit local user groups, like the Linux users group, and give a five-minute spiel about the Regional Info Watch. If you can, make and give out a little flyer that describes the group, has the meeting time, Web site URL, and mailing list information.

Don't forget government agencies, such as county or city – they've got sysadmins too, y'know.

Get some law-enforcement participation. If you don't know any, call your nearest FBI office and ask to speak to an agent who handles computer and/or high-tech crime. Some offices have a squad specializing in that, while others just have one or two agents who handle those areas. Call the local police and sheriff's departments, too. Some places have people dealing with high-tech crime (we've got an entire task force here).[11] Oh, and there's also the prosecutors – district attorneys, state attorneys, and US attorneys. And it never hurts to ask each of them if they know people at other agencies who might be interested.

The Meeting

So you've got a meeting scheduled and people invited. What now?

About two weeks in advance, verify with your speaker that they're coming. Don't rely on the speaker to remember to contact you if they have to cancel, especially if they have something like a family emergency.

Send out a meeting announcement to the mailing list at least a week in advance. Include the time and date, location, directions, and a description of the talk. If you want, send out a (short) reminder the day before, too.

Before the meeting, make sure that whatever you need for the meeting is in place and working (chairs, video projector, whatever). Also, make a list of things to talk about: news, new vulnerabilities and exploits, tools, etc. Don't rely on your audience to supply information – anything that they do is a bonus.

11. Computer and Technology Crime High Tech Response Team: *http://www.catchteam.org*.

If necessary, put up signs showing people where the meeting is. Also have someone to greet and escort the speaker to the meeting.

Plan to show up for the meeting a half-hour early so that you can be prepared and greet people as they show up. You may have to assist the speaker in getting set up. Put the meeting agenda (see above) on a whiteboard. Also put up the URL for the Web site, subscription information, and when the next meeting is (once it is determined).

One agenda item for your first meeting is to set a regular time for future meetings. We have found that a consistent meeting time is helpful (ours are currently the 2nd Monday of the month). People tend to set aside the time when they know when the meetings are, and they remember a regular time better. Get some consensus on the time – more people show up that way.

Also, see if you can get volunteers to give presentations at upcoming meetings, at least for the second meeting. We pitch our meetings as a "friendly audience," a way to try out a presentation before giving it elsewhere.

Finally, use the social hour as an opportunity to thank people for showing up and get feedback from people on what they liked or didn't like. You may find that people volunteer to help or volunteer to speak.

Lather, rinse, repeat the following month.

Getting Speakers

The hardest part about doing this is rounding up speakers. Sometimes you get volunteers quickly, but often you have to do a little convincing. Sometimes when somebody tells us about something they're working on, we ask them if they wouldn't mind talking to the group about it. Occasionally, we pick a topic and find somebody to speak on it. Other times, we see a presentation elsewhere and invite the speaker to give the same presentation to our group.

We emphasize that the talks can be on a wide range of topics, 20 to 40 minutes, and can be low-level or high-level. In this way we have a forum that attracts both beginner and expert-level people.

And we try to have a balance between technical and non-technical presentations. As it turns out, the legal/law enforcement people are often very interested in the technical presentations, and the geeks are interested in the legal presentations.

But we strongly *discourage* presentations by vendors who just want to do a sales pitch. We allow product vendors who wish to do a technical presentation on their product – how it works and what problem(s) it solves. But in our case, some members of the audience are quite savvy and quick to tear apart a product that is smoke and mirrors – and we point this out to the vendor. Since our mission is to be educational, there's no value in our "members" learning about a tool that is not effective.

Some Tips and Gotchas

Some random tips, in no particular order:

- Be sure to coordinate with your speaker on what technical requirements they have for their presentation (e.g., video projection, Internet connection, audio, microphone) and/or what you are able to provide. And make sure that someone is present who knows how to work the equipment.

- If applicable, on the day of the meeting inform your receptionist(s) about the meeting so they can direct people to it.
- If you have to change a meeting from its regularly scheduled date or time, send out extra announcements making it clear that it's not at the usual date/time.
- Remember to provide directions and parking information. No use having a meeting if people can't find it.
- The meeting date/time has a big effect on who shows up. Some people do better with mornings (law enforcement), some do better with afternoons (geeks). Some people prefer after-hours, but others want to be home with their families (or don't want to miss the latest episode of "Buffy the Vampire Slayer").[12] Friday afternoons are probably the worst day for a meeting, as many people take off early Friday to do weekend stuff or are caught up with end-of-the-week tasks that they need to finish.
- Consistency and continuity is important. Keep the meeting cycle going. If once a month is too much, try once every two months, but keep it steady. If your meetings become too irregular or infrequent, attendance will taper off drastically.

Finally...

If you do form a Regional Info Watch, let us know, and we'll put a link to your site on our Web page. And we'd love to know how it's going. Just send an email to *sdriw@sdriw.org*.

Oh, and if you'd like to give a presentation at SDRIW, send an email to *speakers@sdriw.org*. We're always looking for speakers.

12. *Buffy the Vampire Slayer: http://www.buffy.com/*.

secure from what?

by Peter H. Salus

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

peter@matrix.net

The 11th Security Symposium took place in the San Francisco Marriott, August 5–9. I was struck by several things: how much interest there was in the concept of "security," and how little that concept had changed since 1988, when I attended the First Security Workshop in Portland, Oregon.

The keynote this year was by Whitfield Diffie, the co-inventor of the Diffie-Hellman protocol for public key cryptography a quarter-century ago. Whit posed the question, "What is security?" and responded that it was "prevention of adverse consequences from illegitimate acts of human beings." Not bad. But it struck me, as it has before, that we spend too much time considering intrusions and break-ins, on encryption, passwords, and firewalls.

Back in 1998, Bill Cheswick gave a wonderful talk at SANE in Maastricht, involving castles with moats, the Great Wall of China, siege warfare, and sneak attacks.

Nowadays, tens of thousands of enterprises depend on the Internet. They are as vulnerable to cutting off that service as ancient and medieval cities were to sieges. Preventing access to their customers and suppliers would be as disastrous to the modern corporation as lack of access to food and water was to the besieged. And this is exactly what SYN floods and DDoS attacks do: they render Internet communication impossible to the besieged.

Diffie pointed out that the goal of security is not retaliation, but denial of success. Thus, preventing others from obtaining secrets is important. Losing secret information is a vulnerability. But so is the inability to use information. The proliferation of the Internet, Diffie pointed out, leads to increasing power, increasing digitization, and a need for security remote from the individual.

There was much more in Diffie's keynote, and there was a lot of food for thought. Which may be yet more important.

The Plan 9 guys gave a great paper on security in Plan 9 resulting from constraints on privileged execution of server processes through "factotum." If you weren't in SF, read the paper.

I then toddled off to hear Ed Felten of Princeton speak on the "Freedom to Tinker," the law's impact on technologies being of increasing concern. Felten understands the freedom to tinker as vital to our ability to understand, discuss, repair, and modify the technological devices that we own.

When I was a kid, I liked to take things apart: an alarm clock, a toaster, etc. I'll bet most of us did. Taking things apart (and, as time went on, putting them back together) brought about understanding. I'm willing to state that my ability to explore and understand has, over time, benefited others, too.

Felten made the point that "Tinkering is socially important." He's right.

On Thursday, I went to listen to Stuart Staniford, Vern Paxson, and Nicholas Weaver talk about the ways that script kiddies wreak mischief. While valuable, I found the most important point of the paper buried at the end: while he has advocated it in the past, Paxson makes an increasingly salient point in advocating a sort of CDC for cyberspace, an organization whose mission is to track and monitor various forms of cyber-disease and attack.

With the attention being paid to "homeland security" these days, $10 to $25 million a year to map, track, dissect, and analyze the matrix seems almost trivial.

Paul Kocher's "Illusions of Security" had a number of good points embedded in it: security does not equal functionality; "Most commercial products have negligible probability of being very secure against creative attackers"; "There are too many people designing secure systems who have never broken one" (refer to Felten's right to tinker); and "There is a shrinking ratio of engineers to problems."

On Friday morning, I listened to Pam Samuelson's excellent presentation on the DMCA. Again, Felten came to mind. While Jack Valenti and the TV/video, music, and film industries may have a lot of money, they're wrong-headed. Preventing reverse-engineering is like passing a law against taking apart an alarm clock.

I keep having this bizarre ahistorical fantasy in which Petrarch sues Chaucer and Milton and Shakespeare for "reverse-engineering" the sonnet.

Lots of things to think about, hence a fine conference.

the bookworm

by Peter H. Salus

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

peter@matrix.net

December already!

I know that all of you are eagerly awaiting Christmas/Chanukkah or something. So I thought I'd begin with two items you might want someone to buy for you.

Gift Suggestions

First, a *perfect* gift for anyone who works with/on the Internet: Addison-Wesley has brought out a boxed set of *TCP/IP Illustrated* – three volumes by Rich Stevens (and in vol. 2, Gary Wright) that are invaluable. The boxed set comes with a poster of the 4.4BSD TCP/IP Networking Data Structures. I tacked mine above my desk. While the set runs $169.95, that's 15% less than buying the volumes separately. I've seen it discounted, too. ISBN 0-201-77631-6.

The second is far smaller and cheaper, but it's worth reading and thinking about. This is *Free Software, Free Society: Selected Essays of Richard M. Stallman*. At just over 200 pages, it's a "lightweight," but it's not at all light conceptually. The contents are divided into four sections: history and philosophy of the GNU Project and free software; the politics of copyright and patents; freedom, rights, and the threats of globalization and proprietary software; and "The Licenses" – containing the GPL, the "Lesser" GPL, and the "Free Documentation License." Larry Lessig has written an Introduction to the volume. Whether you're a GNU lover or not, these are singular essays; no one but RMS could have written them. And even when RMS is at his most irritating, what he says is important and valuable.

Command Guide

Last column I lauded the *Universal Command Guide*. I've learned that there's a Web site: *http://www.allcommands.com*.

It is a Web version of the command finder which helps you find the command in any operating system by typing in the task you want to perform. Guy Lotgering also informs me: "We will also be putting the entire book in a searchable online database available on my Web site."

Really laudable.

Reading and Reference

Once you've read and internalized the 2000-plus pages of Stevens, you'll want to move over to Pike's volume on network security. Pike covers the PIX firewall; the implementation of IPsec in the Cisco environment; etc. I enjoyed reading Pike's exposition, but I was irritated by the sparseness of his references. Encryption is limited to Diffie-Hellman, Stallings, and Schneier. Talking about firewalls without a mention of Bellovin is bizarre. You'll search in vain for many other notables, too.

Staying within networking but broadening your horizon beyond Cisco, you should look at Edwards et al. This book provides you with solid bases for examining the rationale and benefits of multicasting on the Internet. I happen to think that multicasting is an important technology, but I'm unconvinced that it is the killer app some people think it is. The descriptions of the protocols are fine. And this book has a bibliography.

VoIP

I'm not sure whether VOCAL (Vovida Open Communication Application Library) is the "answer" to VoIP, but it's certainly a good beginning. Two of the authors of *Practical VoIP* are at Cisco, and Cisco sponsors the Web site, *http://www.vovida.org*, but VOCAL is an open source product. Calling, routing, billing, etc., are all covered in this book. It's another example of the excellence I've come to expect from O'Reilly.

Revisiting

Bradley's second edition of his XSL volume is quite excellent. XSLT is a companion to XML, one of the "standards" promulgated by the W3 Consortium. If you work with XML and stylesheets, this is a definite must have.

Top Ten for 2002

The Stevens trilogy doesn't qualify for this year's top ten, as the volumes were originally published too long ago, but Stallman's volume certainly belongs here. (As I say each year, these ten are not in any particular order.)

See you all next year!

Evi Nemeth et al., *Linux Administration Handbook*. Upper Saddle River, NJ: Prentice Hall, 2002. 889pp. ISBN 0-13-008466-2.

Richard M. Stallman, *Free Software, Free Society...* Boston, MA: GNU Press, 2002. 224pp. ISBN 1-882114-98-1.

Thomas Sterling, ed., *Beowulf Cluster: Computing with Linux*. Cambridge, MA: MIT Press, 2001. 496pp. ISBN 0-262-69274-0.

Nick Christenson, *Sendmail Performance Tuning*. Boston, MA: Addison-Wesley, 2002. 228pp. ISBN 0-321-11570-8.

Martin McCarthy, *The Procmail Companion*. Edinburgh, Scotland: Addison-Wesley, 2002. 235pp. ISBN 0-201-73790-6.

book reviews

Brian Ward, *VMWare*. San Francisco, CA: No Starch Press, 2002. 249pp. ISBN 1-886411-72-7.

Robert J. Chassell, *Introduction to Programming in Emacs Lisp*. 2nd ed. Boston, MA: FSF, 2001. 292pp. ISBN 1882114-43-4.

Gary McGraw & John Viega, *Building Secure Software*. Boston, MA: Addison-Wesley, 2001. 528pp. ISBN 0-201-72152-X.

Guy Lotgering & the UCG Training Team, *Universal Command Guide*. New York: Hungry Minds, 2002. 1591pp. + CD-ROM. ISBN 0-7645-4833-6.

Michael Lucas, *Absolute BSD*. San Francisco: No Starch Press, 2002. 616pp. ISBN 1-886411-74-3.

Four more that are almost as good:

James Pike, *Cisco Network Security*. Upper Saddle River, NJ: Prentice Hall, 2002. 302pp. ISBN 0-13-091518-1.

Brian M. Edwards et al., *Interdomain Multicast Routing*. Boston, MA: Addison-Wesley, 2002. 356pp. ISBN 0-201-74612-3.

Luan Dang et al., *Practical VoIP Using VOCAL*. Sebastopol, CA: O'Reilly & Associates, 2002. 502pp. ISBN 0-596-00078-2.

Neil Bradley, *The XSL Companion*. 2nd ed. Harlow, UK: Addison-Wesley, 2002. 465pp. ISBN 0-201-77083-0.

THE QMAIL HANDBOOK

Dave Sill

Berkeley, CA: APress, 2002. 492pp. ISBN 1-893115-40-2.

Reviewed by Dustin Puryear
dustin@puryear-it.com

A popular MTA that has been ported to various UNIX flavors as well as FreeBSD and Linux, qmail was written with security and scalability in mind. Indeed, the software has a record for being rock hard in terms of resiliency to attack and its ability to cope with high loads of mail traffic.

The power and reliability of qmail is great news for the harried network administration staff, but having the qmail software alone does not solve the problem of learning how to best configure and deploy it. That's where *The qmail Handbook* steps in.

The qmail Handbook is written as both a technical introduction to the workings of qmail and a how-to manual detailing the installation and configuration of the software. The book is broken into 12 chapters, with six appendixes, and Sill uses first person throughout to create a book that is quite readable.

He begins with a basic overview by covering architecture and some of the details of the inner workings of qmail. Unfortunately, Sill doesn't really go into as much depth in this chapter as I had hoped – I was looking for more background information about the design decisions made by Bernstein, the author of qmail, but that information was not generally forthcoming. Rather, much of the first chapter is a basic and pretty generic MTA tutorial. Sill rectifies this lack of detail somewhat by discussing the qmail architecture in more depth in later chapters and in Appendix A, "How qmail Works."

As the book progresses, Sill covers installation and basic configuration issues and includes a wide range of important topics such as relaying and virtual domains. By the fifth chapter, Sill has already described how to manually inject messages, provided an overview of each file available for use (qmail is very file-centric in how it is configured), and walked the reader through basic methods for controlling qmail and the qmail queue.

I have one small criticism of Chapter 3, "Configuring qmail: The Basics." Sill does provide a good review of all of the configuration files, but instead of grouping file descriptions into related categories, he simply uses alphabetical order. I would like to see more thought put into grouping the file descriptions into related categories so that readers can quickly jump where they need to go. For example, readers may want to be able to quickly review all files related to virtual domains. Currently, you will need to skim all file descriptions to be sure you catch everything you need.

One of the most helpful chapters for most readers, especially those new to qmail, will be Chapter 6, "Troubleshooting qmail." Sill does a good job of running through various troubleshooting scenarios, and details which tools should be used for each job.

The next several chapters are devoted to covering popular topics, such as sendmail migration, performance issues, junk mail, mailing lists, and accessing mail from remote systems using POP3 and IMAP. In addition, a chapter devoted to virtual domains is included that describes the use of VmailMgr and Vpopmail, two "add-ons" to qmail that make virtual domain support easier.

In the final chapter, Sill tackles some of the more difficult issues, especially those that face administrators of high-volume mail servers. In addition, he reviews customizations of qmail to allow antivirus scanning, patching the source to allow use of a database backend to store user account information, and more.

book reviews

Sill also includes valuable information in the six appendixes: "How qmail Works," "Related Packages," "How Internet Mail Works," "qmail Features," "Error Messages," and "Gotchas." Most users new to qmail will certainly appreciate "Related Packages," which contains a list of software that can be used to add features to qmail.

I found Dave Sill's book quite clear and full of good information. Alas, like many books written about specific software, *The qmail Handbook* lacks sufficient focus on management issues. Specifically, I would like to see more information on account management and more in-depth coverage of database back ends used for authentication and perhaps even mail storage (although Maildir works quite well in an NFS environment due to its excellent design) and qmail used for large, clustered environments that handle very large volumes of mail. Certainly Sill addresses some of these concerns, but mostly in a piecemeal fashion. These minor issues notwithstanding, this is a good book for those seeking a solid understanding of how to install, configure, and use qmail.

RUBY IN A NUTSHELL

Yukihiro Matsumoto w/translated text by David L. Reynolds, Jr.

Sebastopol, CA: O'Reilly & Associates, 2001. 204pp. ISBN: 0-596-00214-9.

Reviewed by Raymond M. Schneider
ray@securityfoo.net

Here we go, it's a Ruby revolution. O'Reilly has come out with another text on the jewel scripting language from Japan. Last time around the book was hardly more than a pamphlet, a pocket reference in Japanese. This time it is a Nutshell text. It would seem to the innocent bystander that Ruby is moving up in the world. A quick grep of one of the book-selling Web sites shows at least five texts dedicated to the Ruby language. Ruby is getting mature.

Ruby in a Nutshell is like all the other O'Reilly nutshells: it's a light desk reference. The reader (most likely a programmer or system administrator) should be familiar with programming on at least a novice level. *Ruby in a Nutshell* is broken down into four distinct sections: language basics, built-ins, standard library, and tools.

In the Language Basics section the book follows the format of the nutshell series, giving the reader an extremely brief overview (30 pages), but it serves its purpose. Readers familiar with other scripting languages will notice some inferred opinions. For instance, the author writes in a section entitled "Whitespace," "We'll leave the thorny questions like 'How much whitespace makes code more readable and how much is distracting?' for another day." The reader then discovers that Ruby ignores whitespace. This is sort of poking fun at another scripting language, python, which does make whitespace count. The author's expressed attitude throughout *Ruby in a Nutshell* is lighthearted and fun.

The Built-In Library Reference section explains exactly what a programmer would expect to find in this section: predefined globals, built-in objects, and their methods. This section totals 73 pages in length and is terse in explanations.

The Standard Library section contains those things that are essentially extensions of the Built-Ins. In order to use these libraries a programmer must "require" them in the code. The reader will find things about Basic sockets and TCP sockets here. This section also includes information about CGI and a decent, though short, example.

The final section,Tools, talks about things every programmer using or thinking about using Ruby will want to have around and know how to use. For instance, the Ruby debugger is addressed here, as is a profiler, a code tracer and interactive Ruby. One of the last things here is "ri." Ri is to Ruby what perldoc is to perl.

Ruby in a Nutshell delivers a quick and dirty reference for an incredibly fun language to program in. The organization of the book makes sense, so the programmer can quickly find the needed reference. It is a joy for me to read a short (204 pages) book about a programming language that contains a usable reference. *Ruby in a Nutshell* never fails as a quick reference for checking syntax or checking for the right method of an object class.

There are only a couple disappointments with *Ruby in a Nutshell*. The first is that the Ruby C API is not covered. This is disappointing, as extending and embedding Ruby is something that happens quite often. The second is that terse explanations sometimes require the reader to seek out more complete information elsewhere. It is unfortunate that O'Reilly has not given us a full-fledged Programming Ruby text. I hope that's around the corner.

If you are looking for a desk reference, *Ruby in a Nutshell* is for you. If you want to learn the Ruby language, this is definitely not the text you need. I recommend *Programming Ruby: The Pragmatic Programmer's Guide* (Addison-Wesley, 2000, ISBN: 0-201-71089-7) for that purpose.

USENIX news

Nearly 20 Years Ago in USENIX

by Peter H. Salus

USENIX Historian

peter@usenix.org

The date on this issue of *;login:* is December 2002, but the next issue will be February 2003, and I want to write about January 1983, San Diego, CA. About 1850 eager UNIX users gathered for "UNICOM" – a combined meeting of STUG (Software Tools User Group), /usr/group (in later years UniForum), and USENIX.

This was the meeting where Bob Guffy announced that AT&T was about to introduce System V; where Bill Munson announced that "DEC supports UNIX"; and where Bob Fabry announced that 4.2BSD was "almost ready" (and that it was 46% larger than 4.1: 53,500 lines of code!). Hot stuff!

There were two divergent things I noted as I re-read the Proceedings: (1) how many of the topics (and companies) were passé and (2) how many of the people (and papers) were notable.

Passé as well as notable. It seems bizarre. But here's a paper on 4.2BSD on the Sun Workstation, by Tom Lyon and Bill Shannon. And one on porting 4.1 to the λ750 VLSI, by Paul Chen and Chet Britton. And, truly notable, John Chambers and John Quarterman's paper comparing System III and 4.1BSD.

But that's a mere beginning. There was Eric Allman on "Mail Systems and Addressing in 4.2BSD"; Mark Horton on "Usenet"; Mike Karels on *vfork*; Brian Harvey on "UNIX Logo"; Mike O'Dell on "Portability"; and Mike Tilson, Jean Wood, Joe Yao, John Mashey, Jim Isaak, Heinz Lycklama, and . . . , and . . .

But there was more to reflect upon.

First among these was a paper by Jim Lawson, "UNIX Research at Lucasfilm." We're talking about January 1983. At that time ILM had already won awards for *Star Wars* (1977), *The Empire Strikes Back* (1980), *Raiders of the Lost Ark* (1981), and *ET* (1982), and were soon to receive an Oscar for *Return of the Jedi*.

Lawson talked about running four VAX 11/750s, one 11/780, and a network of 6800s. He also remarked on the fact that Lucasfilm was awaiting 4.2BSD support. And he said that Lucasfilm has found UNIX to be the "ideal operating system."

Really impressive.

On a similar topic, Jeff Loomis and Phil Mercurio talked about "Computer Animation at UCSD."

The other topic that struck me was standards. While /usr/group had begun its UNIX standardization efforts at UNICOM in 1981, now Heinz Lycklama announced the availability of a "draft UNIX interface standard." Jim Isaak spoke about "Standards Organization"; Rob Petersen gave a paper on "The History and Purpose of Standards"; and Robert Swartz talked about "Criteria for Standards."

Twenty years later, many issues of *;login:* have a section on the POSIX committees. Sigh.

There were many other interesting papers, but what strikes me as I write is that so many of the things that seemed important (UNIX on the NS6032, porting to the Gould 32/27, UNIX on Apollo computers) are meaningless today.

Oh, by the way, there was a special session on "Marketing and Venture Capital." How little we knew.

IOI 2002

Yong-In, Korea

by Don Piele

USACO Director

piele@cs.uwp.edu

Sunday evening, August 18, 2002, the US delegation arrived in Korea and were bussed to Kyung Hee University, the site of the International Olympiad of Informatics for 2002. Our delegation consisted of thirteen members, including a pair of international committee members, two team leaders, two observers (to learn how to run the IOI in the USA in 2003), four contestants, and three visitors.

The opening ceremony was held in Renaissance Hall in the Central Library for 277 contestants along with a host of leaders and guests, for a total near 600. It began with a short report on IOI 2002 followed by two messages, one from the president of IOI 2002 and another from the Minister of Science and Technology for Korea.

The contestants were sent off to their rooms by 8 p.m. while the team leaders approved the three problems for the first competition day and began the long task of translating them into their native language (since contestants are not required to know English). Some did not finish until the wee hours of the morning.

On the first competition day, the contestants were given five hours to work on the three problems of round one. The word after the time was up was that they were a very challenging set of problems.

Thursday, the second set of problems was appreciated by the competitors more than the first. Everyone came out feeling relieved it was finally over.

When we got the results back they were all very close together – around 200 points out of a possible 300 points on round 2. On the first day, Tiankai Liu had a higher than average score because of his success on one problem which he solved better than all other competitors at IOI. His total number of points (415) for the two days was over a hundred points ahead of the rest of our team. The other three, Jacob, Adam, and Alex, were close together with totals just under 300 and combined with day 1, good enough for medals.

Overall, the team did a wonderful job getting medals in a very difficult competition. The problems used by the Koreans were very original and forced the students to think outside the box. Tiankai showed his extraordinary ability to do just that by coming up with an original solution that surprised even the judges. He was the only one in the competition to get full marks on the problem called XOR. His winning solution was only 100 lines of code, about a third as long as expected, and it outperformed the judges' solution. Remember, this is only one of three problems he had to solve in five hours and the judges had spent weeks finding their best solution.

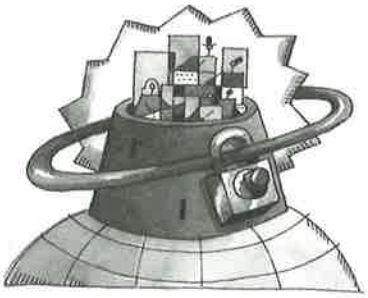
Tiankai Liu, a sophomore from Phillips Exeter Academy in Exeter, NH, captured a gold medal. The other three members of the US team won silver medals: high school seniors Adam D'Angelo, also from Phillips Exeter Academy, Jacob Burnim, from Montgomery Blair High School in Silver Spring, MD, and Alex Schwendner, a home-schooled freshman from Austin, Texas.

Burnim captured the highest silver medal during the competition. Schwendner, a freshman, was the youngest team member from the US ever to receive a silver medal. Tiankai had the fourth highest score in the whole contest and this was his first year at IOI.

In addition to his gold medal Liu was awarded the Sens Q prize, newly established by Samsung Electronics for the contestant who solved the problems the most creatively. He received a Samsung laptop computer as his prize.

Participation in the IOI 2002 was made possible by USENIX, our sponsor for the USACO.

The team: from l. to r.: Jacob Burnim, Alex Schwendner, Tiankai Liu, Adam D'Angelo



conference reports

OUR THANKS TO THE SUMMARIZERS:

Akshay Aggarwal
Mihai Christodorescu
Michael Hohmuth
George M. Jones
Lou Katz
Prem Uppuluri
Haining Wang
Seung Yi

11th USENIX Security Symposium

SAN FRANCISCO, CALIFORNIA, USA

AUGUST 5-9, 2002

KEYNOTE ADDRESS

INFORMATION SECURITY IN THE 21ST CENTURY

Whitfield Diffie, Sun Microsystems

Summarized by George M. Jones

The opening keynote given by Mr. Diffie provided a jam-packed overview of information security in the 20th century and projections for the 21st century, interspersed with sage opinions and observations.

He began by defining security as (1) preventing adverse consequences from illegitimate actions of human beings; (2) protecting yourself against the actions of an intelligent opponent; and (3) something that gives you the appearance of legitimacy.

The history of information security in the 20th century was largely dominated by issues of privacy, with cryptography being the primary tool to enforce privacy. While cryptography has been around at least since the days of Julius Caesar, its importance became crucial with the advent of a new communications technology: radio. "Radio revolutionized warfare. Before radio, a naval fleet commander sent ships out with orders and could communicate with them every few weeks or months at best. With the advent of radio, orders could be communicated in, at most, days. *But* since radio is a broadcast medium, everyone could listen – hence the importance of cryptography for confidentiality."

Accordingly, WWI saw an increase in the use of cryptography, and WWII, an increase in the use of automation (the code clerks just could not keep up). Ensuing decades brought computerization ('50s); reliable computers, timesharing, and the first "computing communities" ('60s); the advent of the ARPANET and the loss of all the (dis)advantages of locality ('70s); disconnected, password-less PCs ('80s); and the Internet brought to all those password-less PCs (oops), the "computerization of everything," and the migration away from paper ('90s).

Another important focus of information security in the 20th century was the efforts to provide secure voice communication, from the '40s, when exactly two people (Roosevelt and Churchill) could communicate securely using multi-million-dollar 30-ton devices, to the STU phones of the '80s, which "reached their goals but failed because communications expanded beyond the phone (cell phones, voice over IP, PDAs, fax, email, WWW)."

Diffie outlined some trends and observations from the 20th century: computer power keeps increasing, information is now digital, security technology moves closer to the user, DES was developed in secret, AES was developed in public. In the shift to elliptic-curve cryptography, "we're now moving from using 17th-century to 19th-century mathematics." Encryption allows networks to be defined by who has what keys, not by topology (à la today's firewalls). "The minute you begin rolling out crypto, you turn everything into 'us' vs. 'them'."

Some current trends: computer-mediated communication, the rise of the information economy, unification of communication and delivery channels (e.g., Web site download of programs), mobility, and bandwidth on demand. "*The* driving factor is that better security technology draws more valuable traffic, and, conversely, more valuable traffic requires better security."

Diffie had some insightful observations on privacy: "Privacy is a security policy

about personal information. If you don't have any way of controlling information flow, you have no way of enforcing a policy. There is an increasing immediacy to information security. It's important that people be able to recognize each other (authentication) and have private conversations (confidentiality). We are trying to transplant our human culture into a world of computer-mediated communications."

On the "open" vs. "closed" development approach, he noted that "Some argue against this, saying 'open' means the 'bad guys' can look at it. Some argue for it, saying many eyes mean more security. They both miss the point. 'Open' means *you* can look at it and satisfy yourself."

In answer to the question "Why is it taking such a long time to get a working PKI?" he noted that "most of the costs are up front, but most of the benefits accrue once it's deployed. Unlike PCs, it's hard to deploy PKI piecemeal.

"Key escrow is like the One Ring in the *Lord of the Rings*. It is an evil that will be back . . . though perhaps under different names." Data recovery keys are valuable to data owners to ensure the ability to recover private data.

"Today, flows of information are controlled by the movement of people, e.g., it's cheaper to hire away a Microsoft employee than to gain certain information by other means."

Security is about people. "It is never independent of point of view. It often deals with competing interests. It is never value-neutral. It moves power from one group to another."

In closing, Diffie offered the following thoughts: "Security should make doing business easier, not harder. *Nothing* is more important than human-factors engineering. Quality of security is directly proportional to quality of evaluation." Simplicity is the essence of evaluable security design. There are big gains to be had from putting some functions in hardware.

"The question for the 21st century is, 'Can everyone be secure at the same time?'"

Questions and Answers

Q: (Steve Bellovin) Most of the problems we're seeing are not crypto problems e.g., buffer overruns, etc.

A: The fact that we can't implement things right is our Achilles' heel.

Q: (John Ioannidis) What about attempts to legislate security out of existence?

A: My prejudices agree with yours. Security is just one piece in a larger puzzle. Information is becoming a commodity. Societies have always regulated commodities. Decisions made today will shape society for decades.

INVITED TALKS

Wireless Access Point Mapping

Simon D. Byers, AT&T Labs–Research

Summarized by Akshay Aggarwal

Wireless is appearing almost everywhere and comes with no strings attached, literally. Simon Byers spoke about his experiences in wireless access point (AP) mapping. He started off by pointing out the pervasive nature of 802.11b-based wireless LANs, stating that they could be found in your neighborhood McDonald's, Trader Joe's, or just about anywhere. Many laptops now come with built-in support for these networks. "Executives love it," said Byers while illustrating that it was as easy to use wireless LANs in corporate boardrooms as in the female restroom of AT&T Labs' Florham Park, NJ, facility where he works. This property fields wireless LANs as a probable ISP medium, which could solve the last-mile problem.

The wireless LAN protocol uses the free 2.4GHz range and cannot penetrate stones, leaves, or people, though Tupperware is unable to stop it. At a speed of 11Mbps, wireless networks are fast and can be used as access points, relay, or point-to-point links and are "minimally invasive apart from heating up the children a bit." Satellite networks, the only alternative to wireless networks, suffer from poor upload rates and high latency problems. Speaking about the "security" of 802.11b, he pointed to the plethora of literature and scripts available for anyone to break WEP. The end result, according to him, is that WEP is now next to worthless.

According to Byers, open networks exist with the aim of providing free Internet to the people. Some important issues with open networks are the pushback from ISPs, with cable companies persecuting NAT users, patchy coverage, and their susceptibility to DDoS attacks. The reasons to map these networks included the need for a security survey, to find an open network to connect to, to provide and assess network coverage, and to explore the saturation of the free spectrum. To emphasize his point, he gave the example of a war-driving contest at a recently concluded hacker conference (and the basic flaws in the contest). He then showed slides of his mapping efforts made while driving around Las Vegas and New York.

The audience was acquainted with the hardware needed for WAP mapping. He showed them 802.11b wireless cards, the various kinds of antennae (yagi, omni, panel, and dish), and GPS systems and amplifiers. Then came a tutorial on how to build a base station and receiver to capture images from X10 wireless cameras, deployed with the tagline, "You'll never know what you will see!"

While he was driving around Manhattan he found approximately 4000 access points. Of these, 964 had WEP enabled, 156 networks had the default SSID, and many had their addresses as SSIDs. In correlation with other data, analysis of this data provides the location of APs and the comparative reach of the networks. Techniques used to locate APs include max signal-to-noise ratio, triangulation, intersecting spheres, or just the

plain old telephone book in cases where SSID was an address.

To conclude his talk, Byers discussed the business application for mapping APs, which were to set up, manage, and analyze a network for use by all. This included optimizing the deployment and mapping the target customers' footprint. This information would be invaluable to owners of wireless networks.

Freedom to Tinker

Edward W. Felten, Princeton University

Summarized by George M. Jones

Professor Ed Felten of Princeton spent some time this year thinking about the legal and economic aspects of "The Right to Tinker." This follows the presentation last year of his paper on the SDMI challenge (detecting/removing digital watermarks on audio samples), which followed a lawsuit backed by USENIX and EFF to defend his right to present it. This talk outlines some of his conclusions.

"A funny thing's happened in my career," he began. "I've gotten involved in legal issues, or to put it more accurately, those issues have gotten involved with me. Things computer science people have always done are increasingly at risk of becoming illegal. Tinkering benefits everyone, not just techies. We need to sell the idea that the public will lose out as the freedom to tinker is eroded."

"The freedom to tinker," Felten said, "is the freedom to understand, discuss, repair, and modify technological devices that you own."

Felten said that three points need to be stressed:

1. Tinkering is socially important.

Tinkering is rooted in the basic human need to explore and understand the world around us and to control our surroundings. Imagine laws making it illegal to fix your own car. Tools are important to tinkering. Sledgehammers and program debuggers have legitimate and illegitimate uses. Laws such as the Digital Millennium Copyright Act (DMCA) are making useful tools illegal without regard to potential legal uses. Tinkering with products by security researchers benefits the public by disclosing flaws in products they rely on. Tinkering benefits vendors by giving them the opportunity to fix the flaws.

"Increasingly, technology [computers] is controlling access to content. It's no longer just you and the book. Now it's you and your Web browser, and Google, and thousands of Web servers backended by databases connected to networks," said Felten. Tinkering with these technologies should be protected.

Public policy debates often turn on the understanding of technical issues: for example, is a large software vendor simply designing more efficient programs or programs intended to limit competition? Tinkering by independent analysts raises understanding and thus raises the level of public debate.

2. Tinkering is economically efficient.

Most arguments against tinkering boil down to economics, but it is not clear that the arguments are valid when applying generally accepted principles of economic analysis.

Tinkering has many positive side effects (or "externalities"). They include innovation, education, and competition.

If there are barriers to tinkering, such as the DMCA or restrictive End User License Agreements (EULAs), not enough tinkering will occur and the positive side effects will be missed.

3. Tinkering doesn't conflict with "intellectual property."

"Intellectual property is not a single thing [under US law]. It is a combination of copyright, patent, and trade secrets."

Trade secrets protect secret material, but only against disclosure by improper means (bribery, threat, theft). It does not protect what is learned through tinkering or "obvious" things such as hair color.

Copyright and patent are intended "to promote the progress of science and the useful Arts," to maximize total (not individual/corporate) wealth, and to prevent outright copying of a product, but not to prevent study or discussion. None of these should present a barrier to tinkering.

"Our opponents say that the battle is between people who are pro-copyright (them) and anti-copyright (us). We don't have to accept that. Our position should be that we respect the traditional scope of copyright; fair use is important but is not the issue. Laws such as the DMCA do harm to people (tinkerers, the general public) who have no intention to violate copyright. It's about maintaining robust, open, competitive technology."

For more info, see *http://www.freedom-to-tinker.com*.

Questions and Answers

Tinkering was needed to facilitate the first question; the audience microphones didn't work. The techies present fixed them, with no help or permission from the vendor or Congress.

Q: What alternative is there to the DMCA (technological or other)? What can we do to prevent/deter infringement of copyright?

A: The DMCA is the worst of both worlds. It does not prevent infringement and punishes those who have no intent to violate copyright. It goes beyond what is needed to prevent infringement. The main effect of the DMCA has been to cause collateral damage.

Q: Would you be in favor of building a tool whose sole purpose is to circumvent infringement?

A: No.

Q: Does this change things from civil to criminal? What about the standard of evidence?

A: DMCA increases the number of parties who can bring a suit. Anybody who is harmed can bring suit. This is the source of a chilling effect. You as a researcher don't know who might be offended/harmed/bring suit. The incentive is to do nothing (not to tinker).

Q: Are there some circumstances where anti-tinkering terms of use can benefit users? For example, pop-up advertising paying for free network access. Is it OK to prevent tinkering to turn off pop-up advertising?

A: If your question is how to change the law and policy to encourage tinkering, this (repealing the DMCA) is the only way.

Q: Napster did have legitimate uses.

A: Napster had too much of a role in the infringement.

Q: Do you have suggestions of practical things that people can do?

A: Participate in forums such as this [USENIX]. Try to influence/talk to reps. Get involved with EFF. Be vocal.

Q: What about obfuscation that raises the cost of tinkering?

A: What's really dangerous are mandates that require people to build in anti-tampering devices.

Q: What about EULAs and the Uniform Computer Information Transactions Act (UCITA)?"

A: UCITA would strengthen EULAs. An important step is to say that licenses should not be used to prevent tinkering.

Q: Do you see any problem with the use of the term "tinkering"? Will people whose primary concerns in life revolve around junk food and big-screen TVs take it as a serious issue?

A: Lots of people like to tinker. Recall the Thomas Edison stories. It's possible to connect the issue to things that concern the general public (e.g., a better way to use your VCR).

Biometric Authentication Technologies: Hype Meets the Test Results

James L. Wayman, Biometric Test Center, San Jose State University

Summarized by Akshay Aggarwal

What exactly is hyperbole? Jim Wayman pointed out that the Merriam-Webster dictionary defines it as an "extravagant exaggeration ('mile-high ice-cream cones')." Much of the hype surrounding biometric identification is just that – an exaggeration of the truth. To illustrate this point further he referred to two Web sites and proceeded to expose their exaggerations.

The first Web site belonged to an unnamed biometric product vendor. Their claims:

1. "Facial recognition technology is the only biometric capable of identifying known people at a distance." This is contradictory to the fact that DARPA is involved in a project aimed at using iris-scanning technology at a distance. Facial recognition is not the only biometric available for long-distance recognition, though it is one of them. In addition, the vendor admits that the range of facial-recognition technology is currently limited to 10 feet. So what is really meant by distance?

2. "Facial surveillance can yield instant results, verifying the identity of a suspect instantly and checking through millions of records for possible matches quickly, automatically, and reliably." Furthermore it claims, "These investigative tools help to single out known terrorists or criminals." This implies that the technology is accurate when, in fact, it suffers from fairly high rates of false positives and false negatives.

The second was a leading educational Web site. Their claims:

1. "The biometrics industry is mythical." The International Biometric Association exists and its members can be found at *http://www.ibia.org*; the industry is far from mythical.

2. "Publicly available, independent evaluation of technologies and products is extremely rare." Independent evaluations and standard testing procedures can be found at sites such as *http://www.biometrics.org*, *http://www.afb.org.uk*.

Wayland says, "Hype is factually correct but leaves an impression that may not be accurate." He agrees with B. Miller's definition of biometric authentication as "automatic authentication or identity verification of a living human individual based on behavioral and physiological characteristics."

Wayman says that some metrics that should be used to evaluate technical performance of biometric algorithms are failure-to-enroll, failure-to-acquire, false positives, and false negatives. Failure-to-acquire measures how often the device fails to recognize a metric, such as when a facial-recognition system fails to recognize a face against a pale background. Failure-to-enroll is a more important metric, measuring whether a biometric precludes certain groups of people; for example, fingerprint scanners cannot be effectively used on the old and the very young, groups that tend to have a much less distinct fingerprint. Thus biometrics cannot be used on all segments of society equally.

Current biometric evaluation involves technology, scenario, vulnerability, security, and operational testing. Cost-benefit analysis, environment testing, human perception response, and user attitude also need to be evaluated in the future. Test results are indicative only of people in a particular environment. A hand geometry system when tested at the Sandia Labs and the nearby Kirkland Air Base produced different results in these two biometric environments.

Wayman was asked about the methods used to search through the large databases; he replied that the databases were usually partitioned on the basis of criteria like gender and, further, on some biometric characteristics. In response to a question about the vulnerability testing of such systems, Wayman pointed out that such tests were needed; he gave the example of the inability of a facial recognition system to differentiate between a human face and a photograph.

In conclusion, he reiterated the long road ahead for biometric devices and research.

Network Telescopes: Observing Small or Distant Security Events

David Moore, CAIDA, San Diego Supercomputer Center

Summarized by Lou Katz

David Moore gave an interesting report on experiments with monitoring remote network events through examination of unexpected packets on some address spaces he monitors. This arrangement, a network telescope, uses a portion of the globally routed IP address space on which little or no legitimate traffic is expected. Monitoring the traffic which does arrive gives a view of certain remote events.

An analogy to monitoring with astronomical telescopes helped convey the operation and properties of a network telescope. In network monitoring, a larger address space increases the "lens size" of the network telescope, as does noncontiguous address spaces. Larger network telescopes can see shorter time durations and lower packet rates, and have a larger field of view with better accuracy for start and end times of events (e.g., Code Red spread at about 10 packets/sec). Both Code Red and global DoS attacks could be seen. The data were collected using a passive tap ahead of the net(s) being monitored.

Attackers spoof source addresses randomly, and it is this "backscatter" that is being seen at the telescope. Analysis of the backscatter could give a quantitative measurement of the DoS. Interestingly, the portion of address space monitored can affect the traffic seen, both positively and negatively, since some types of events attempt to preferentially use address spaces adjacent to their source in order to spread. It is not known how randomly these addresses are chosen. In the initial operation of the network telescope, the deployers of the attacks were unaware of the telescope. Later on, there seemed to be evidence of either deliberate avoidance of the telescope-monitored IP space or of attacks on the telescopes themselves.

Detecting an event is a function of the size of the monitored network. An /8 telescope could detect an attack in a minute or two, while a /24 might take 58 days. A /8 network can track an infection accurately, but a /16 has a time lag and the shape of the curve is wrong. On a log plot, the slope for a /16 is OK but the times are wrong. Work on deconvolving a /16 curve into the /8 curve is being pursued.

Conclusions reached so far: there are lots of attacks; some exceed 600,000 packets/sec. Most attacks are short, but there are some that are continuous for over a week. The attacks don't seem to load the network or major peering points, but some embedded devices (routers, printers, etc.) had servers that crashed and had to be rebooted or power-cycled. A steady stream of new packets into the telescope net has been observed at about 20/hr. These are mostly TCP but there are some ICMP floods and some evidence of ICMP black-holing. Eighty percent of attacks last 10 minutes or less. Attacks seem to happen on a human time scale, with peaks at 5 minutes, 10 minutes, 30 minutes, and 8 hours (human control intervals). The victims are mostly commercial businesses, with minor efforts against home machines. There are odd peaks in .ro and .br space, which may be revenge attacks of one ISP upon another.

Code Red spread has been charted by recording machines sending TCP SYN to port 80 of nonexistent machines. Such sending machines are considered to be infected; the data show 359,000 hosts infected in 24 hours. Characteristics of the infection show that 47% of infected hosts have no reverse DNS; there were 136 .mil and 213 .gov hosts infected. Code Red II, by probing local nets, spreads very rapidly on internal nets. Most of the infected hosts were home/small business machines on cable modems.

The reappearance phase of Code Red was also observed; even though there was lots of press coverage – everyone should have known it was coming back, considerable infection occurred. Daily fluctuations were plotted by rough normalization of the IP addresses to time zones. Interestingly enough, at about 9 a.m. every day in every time zone, hosts come up; activity degraded in the evening and on weekends. A great animated map of the world, which showed the spread of Code Red as growing red splotches, was projected. Really scary to see the world mostly turn red in a very short time.

One of the problems with these measurements is that it is difficult to distinguish computers vs. IP addresses. There were a maximum of 180,000 unique IP addresses infected in a two-hour period but 2,000,000 in a week. There is a DHCP effect over long periods. Old computers get new addresses. So far they have not been able to get a good handle on NAT, and it is hard to get a good estimate.

The author concludes that network telescopes can see and give insight into nonlocal events; you don't have to be there, but small telescopes can't see certain types of small events. This is an example of surveillance without a known purpose or target; data are collected first,

and then you work backward after an event. The slides for this presentation should be available on *http://www.caida.org*.

Illusions of Security

Paul Kocher, Cryptography Research, Inc.

Summarized by Lou Katz

Paul Kocher gave an overview of security evaluated from the point of view of a company, such as his, which is focused on cryptography, and of the problems faced by high-risk commercial systems and big companies. The talk was a review of common problems and misconceptions and an exposition of possible rules to live by.

The standard yardstick for measuring cryptographic security, key length, does not really address the problems posed by real adversaries, who lack the propriety to limit themselves to tidy attacks such as brute force, factoring, or differential cryptanalysis. The crux of the problem is that in assessing the security infrastructure, security implies a zero tolerance for flaws in the face of software developer acceptance of bugs proportional to complexity. Since the testing side of system development can't keep up with the complexity of the products, it is often the case that the front door is strong, but it is easy to break in through the window.

In measuring security one must consider the probability of breaking in vs. the cost of the attack. For commercial products there is a negligible probability of being very secure against creative attackers, especially since systems of exponentially increasing complexity are being created, aided by Moore's Law, but security experts are not compensating by becoming exponentially smarter. Is there an upper bound or expected/mean resistance? What is the risk curve, and against whom are we defending? It is important to evaluate what the resistance against an initial attack might be vs. repeated attacks.

In outlining the characteristics of most flaws, complexity and component interactions were among the obvious dangers. When the evaluation of the security of a software system is to be performed, the goal is either to prove that security of the system is bad by finding a flaw, or lacking that, to do an inclusive analysis to assess the likelihood of additional security problems and to advise whether a product is worth deploying. All the while we are faced with the realities that attacking is easier than designing or verifying; and prevention/testing is hard. A very thorough evaluation is expensive, so the constraints on the evaluation process, time, budget, availability and quality of technical information, and evaluator capabilities, experience, and knowledge of the threat model can compromise the results.

Paul posited that the best work is done before the project is started, by careful definition of the target system's security objectives and a review of the implementation details. A checklist of many single points of failure should be developed (he showed an extensive chart of these) along with a long list of reviewable information, such as the open literature, published specs, network and bus I/O, timing, power consumption, defective computations (errors in computation can be used to compromise keys), error messages, failure codes, examination of disk and memory contents, swap files, and RNG seed data. Even chip imaging should be explored. Of course adversaries might engage in illegal/questionable activities such as dumpster diving, so this must also be taken into account.

What you can include in your checklist is to conduct code reviews, which are useful but boring and hard to do in volume. Code review should include algorithms, usage considerations, and protocol analysis, specific details of which were outlined. The increasing connectedness and complexity in the system, a common source of difficulty, increases weaknesses. Exponential growth in transaction volumes means that unusual transactions are hard to break out by hand and lead to an increased use of computers to do the recognition and analysis.

Security is improved when you design for testability, even though testing is expensive. Security design goals to live by were outlined, and their expense and difficulty were not overlooked. Spend money rationally; don't underspend or overspend on security, hire experienced people, and spend early! Avoid what doesn't work – e.g., design by committee, which is flawed by conflicting objectives and no responsibility. Utilize committees later, as they seem to be fine in keeping a design alive after it is done.

Future directions for improving security focus on people. Vendors need to be convinced to spend on prevention. Weak systems, which allow profits from fraud, will lead to more crime, which will fund more crime. Something needs to be done about the moral hazard that there is currently little vendor incentive for security.

Some of the questions focused on the time frame for security resistance to attack – how long after a system is deployed is it usually attacked? (Others may be ahead of them in the attack queue.) Even a flawed system may be stronger than an alternative, or it may not be economically worth attacking compared to others – breakable systems may still be useful.

In summary, this useful talk, rather than providing any specific insights or giving a recipe or checklist for improving security, highlighted many useful and important concepts to consider and evaluate in designing and establishing a system's security.

Formal Methods and Computer Security

John C. Mitchell, Stanford University

Summarized by Mihai Christodorescu

Mr. Mitchell tried to span the existing gap between the formal methods and the computer security communities, because "theoreticians and coders don't talk to each other." The talk described several applications, the different types of formal methods, and their specific strengths and weaknesses.

A formal method is a technique to analyze a system from its description, without putting the system in motion. For example, it means analyzing executable code and trying to ascertain various properties without actually executing the code. In the big picture, formal methods are meant to help to produce good software efficiently: formal methods are precise and automatable, and they usually capture previous experience. There are several current weaknesses: subtleties are hard to formalize and the tools are cumbersome to use. Most of the formal-methods work is now focused on eliminating these weaknesses.

The goal in formal methods research is to reduce the number of unfeasible problems and extend the set of problems and properties that can be checked. Initially, formal methods were applied to hardware verification, as it has a finite number of states. Currently, program verification is the focus of most researchers, but it is not as successful as hardware verification; due to infinite state space, it can only verify simple things about programs. Computer security is itself a subset of type analysis; a well-typed program should not have security flaws.

One of the applications detailed in the talk was the verification of the Java Virtual Machine verifier, which checks Java bytecode after loading into memory and before execution. Since the verifier is the only check performed on the code *before* it runs, it is essential that the verifier itself is correct and implemented according to the specification. To prove the verifier's correctness, abstract instructions were used to reduce the time and space needed in modeling the verifier. The verification of the verifier entailed two phases: verifying the behavior in the verifier specification, and checking the verifier implementation against the specification. In the second step, the research group led by Mr. Mitchell discovered several implementation bugs in the Sun JVM.

Another area of application is protocol security, which looks at simple network protocols (SSL, SSH, authentication, signing) and checks for exploitable flaws. Most of these protocols are fairly simple in their design and involve a limited number of steps. The complexity of unbounded number of states appears when several sessions of the protocol are considered in parallel: the attacker might conduct several parallel sessions and copy messages from one to another. This area of research created several methods, some less formal (cryptographic-based proofs, Communicating Turing Machines) and harder to reuse or automate, and some formal methods (BAN & related logics, operations semantics, automatic theorem proving, symbolic search for an attack, exhaustive finite-state analysis).

Four formal methods were presented in further detail: model checking, multiset rewriting, probabilistic polynomial time, and protocol logic. Model checking was used in proving that contract-signing protocols were fair, noncoercive, and accountable. Examples of such protocols include Asokan-Shoup-Waidner and Garay-Jacobson-MacKenzie.

Multiset rewriting (MSR) is related to mathematical logic and deals with sets of facts known about the system and transition (or rewrite) rules that modify the system and the facts about the system. MSR has a simple tractable model, but it tends to overlook many things, such as initial conditions. On the upside, MSR is accurate: if an error shows up in the MSR model, the error is present in the protocol. This means that MSR can prove security of a protocol up to a certain set of assumptions but that it will not detect attacks that do not follow these assumptions. MSR usually employs a common intruder model, the Dolev-Yao model, that assumes the adversary is non-deterministic and has no partial knowledge (e.g., adversary either has the encryption key or no key at all).

The probabilistic polynomial time (PPoly) formal method applies the concept of observational equivalence: a protocol is secure if the adversary cannot distinguish its trace from a trace of some idealized version of the protocol. This way, PPoly specifies security by comparing the protocol to a zero-knowledge protocol.

In conclusion, formal methods provide very powerful tools for verifying certain security properties. Most useful right now is the checking of a not too complicated property about a not too complicated protocol or piece of code. The goal of formal methods research is to extend the range of feasible analysis, while keeping them automatable.

"How Come We Still Don't Have IPSec, Dammit?"

John Ioannidis, AT&T Labs–Research

Summarized by George M. Jones

The moderator informed us that "John wants this to be a slugfest . . . so reach deep down inside and find your inner Peter Honeyman."

John Ioannidis then told us that we were really getting four or five talks for the price of one: this talk would mostly work as "How come we still don't have {PKI, IPv6, Mobile-IP, DNSSec, secure email}, dammit?"

He started the talk by contradicting his own title: "We sort of do have IPSec . . .

the question is, why isn't anyone using it?" The rest of the talk was structured around a series of interrogatives.

What is IPSec?

IPSec is a network layer security protocol for IP. It means different things to different people. To some, it's just the wire protocols; to others, it also includes key management, GUIs, and tools.

Why IPSec?

IPSec provides end-to-end communications security at the network layer. It addresses authentication, integrity, and confidentiality. It does not address authorization, privacy, non-repudiation, or perfect forward secrecy.

Why network layer?

The network layer is the choke-point. Putting security in the network layers allows both higher and lower layer protocols to use it. "The seven-layer model is a bit of poison left over from OSI."

What are the benefits of IPSec?

Link encryptors become obsolete. IPSec provides link security to applications "for free." Applications don't need to do their own link security. IPSec allows decoupling of security policies and centralization of management.

While IPSec . . .

During the decade-long saga of defining and deploying IPSec, other security technologies sprang up that may not have been necessary if IPSec were deployed. Among these were the Clipper chip (1993), SSL (1995), SSH, firewalls ("bad"), NAT ("very bad"), and layer-4 re-directors.

Why isn't IPSec? Part I

It's taking too long. SSL and SSH removed the urgency. There are many incompatible implementations. There is no agreement on key management. "OK, we'll just deploy it with IPv6. . . ." Other IETF working groups "rolled their own." "It's all a mess."

Where is IPSec?

"Everywhere and nowhere": *BSD, Linux (Free S/WAN), Solaris, Win2K, VPNs, remote access, academic research.

How is IPSec?

The wire protocols are here and work perfectly. IKE still doesn't have interoperability; there are about 8,000 option combinations. There are no standard APIs. Policy support is rudimentary.

Why isn't IPSec? Part II

IKE is too complex to implement. The docs stink. The configuration of key management and policy are smooshed together. There is no good remote key management and distribution. There is no good evangelizing. Ioannidis informed us that "evangelize," in Greek, means "to bring a good message," but do we have a "good message"?

Why isn't IPSec? Part III

We still have problems integrating with RADIUS, Diameter, and Tokens. We don't have a good PKI. Most of the Internet edge is Windows.

<flame-bait>

"Trying to configure IPSec for Windows has been one of the most harrowing experiences of my life, and I live in NYC!!! There is no good command line interface for Windows IPSec. What good is running a secure protocol on an insecure operating system?"

Whither IPSec?

NAT is an abomination. NAT is broken . . . but I can buy a NAT box for less than $100 and plug in lots of hosts with one IP address now. We need to standardize remote access. We need to work on the APIs. We need better configuration management tools – not just pretty GUIs but something that scales to thousands of systems; these tools just don't exist. We need to play nice with other protocols and host routing. We should work on opportunistic IPSec (Free S/WAN). VPNs are going to be the largest user of IPSec for some time to come. Ubiquitous IPSec would challenge the current firewall model by defining "inside" vs. "outside" with keys, not topology (see Bellovin paper of two years ago). But none of this is any use in the face of buffer overflows and viruses. What to do?

Questions and answers ("Let the games begin.")

Ioannidis got his slugfest, thanks to Microsoft (and his own misunderstandings):

(Dan Simon, Microsoft Research) Q: Perhaps the problem is in the wine-glass model. People want to secure things that IPSec doesn't secure. IPSec started out securing everything and wound up securing nothing.

A: Perhaps we need an N-layer shadow security stack with security at each layer . . . but avoid encrypting N times.

(Dave LeBlanc, Microsoft) Q: We are using IPSec on thousands of machines. We find it quite manageable. We're not going around setting it up on every machine.

A: There are these things called standards; maybe you've never heard of them.

LeBlanc: There is a command line interface, RTFM.

A: Send me a pointer. I don't have a language problem.

[Editor's note: Microsoft uses Active Directory to make this work internally. When I asked Ionnidis months later, LeBlanc still had not provided a URL].

Q: Have you heard about IKE2, JFK, other work at IBM?

A: Yes. I'm one of the authors. A smaller protocol with fewer options was one of the goals and results in fewer lines of code, fewer bugs, better security; simplicity of the spec was the driving force. When you see the doc, we hope it will be unambiguous.

Implications of the DMCA Anti-Circumvention for Security, Research, and Innovation

Pam Samuelson, University of California at Berkeley

Summarized by Mihai Christodorescu

Ms. Samuelson presented an overview of the DMCA and its implications for research, focusing specifically on computer security research. The presentation first covered the rules part of the DMCA, followed by actual cases where the DMCA was used, and closed with possible legal alternatives. The DMCA makes illegal the circumvention of technical measures, with several exceptions, and the circumvention of access controls, with no exceptions (not even for fair use). It was noted that Congress enacted the DMCA as a blanket law with exceptions in place, instead of a less restrictive law that would enumerate illegal actions.

The exceptions are very complex and very narrowly defined. The interoperability exception, meant to allow data exchange between programs from various vendors, is present, but with no indication whether circumvention to gain information useful in attaining interoperability is allowed. The exception for cryptographic research imposes several burdens on the researcher: he or she must be a lawful acquirer of encrypted copy, must get permission to research from the copyright owner, and must have a Ph.D.

The DMCA bans the making and distribution of tools that bypass access controls and copy controls, with the exception of reverse-engineering tools necessary for building interoperability. The problem is in determining the boundary between a description of a technique and a tool implementing that technique. It is unclear whether distributing information (through a Web site, for example) on circumventing a given technical measure is "as illegal as" creating and distributing a tool that performs the circumvention. The Ed Felten vs. RIAA case over the watermarking schemes proposed for safeguarding digital music brought up the question of whether presenting a result at a conference is a circumvention device. In the same case, the exception for cryptographic research could not be applied, as watermarks are not usually considered cryptographic research. Thus, Congress might have created an overly narrow exception, but in the current form of the DMCA, it is up to the court to decide what cryptographic research means.

Another point of contention in the DMCA is the definition of access controls. Tools circumventing access controls are illegal to make or distribute. In many cases, the lawyers forced some technical measures to be considered as access control measures, and thus made them illegal to circumvent. For example, the region-coding of DVDs or the encoding of console games for certain markets are technical measures meant to control the market – these measures overreach and prevent owners of legal copies to use them as they wish (a US citizen cannot play games bought in Japan). The effect is not only limiting to users of the technology but also to competing technologies. *Sony v. Connectix* and *Sony v. GameMaster* illustrated how access controls (e.g., country codes) can be used in an anti-competitive fashion to shut down competing products that bypass access controls, even without allowing piracy.

In the various cases where DMCA was applied (*RIAA v. Felten, US v. Sklyarov, HP v. SnoSoft, Microsoft v. Huan, Edelmen v. N2H2, Sony v. Connectix v. Bleem, Sony v. GameMaster, RealNetworks v. Streambox, Universal v. Corley, DeCSS*), mixed results have emerged from the courts' interpretations of the law. On one hand, the courts have decided programs were protected as speech by the First Amendment, regardless of the form of the program (source or object code). On the downside, fair use rules were not considered applicable to tools that allow both good and bad uses (e.g., DeCSS can be used to play legally acquired DVDs on Linux but can also be used to pirate DVDs). What is worse, the interoperability clause was almost forgotten, with the access control rules overriding any exception – this can lead to legally sanctioned control of data formats.

While it is not the "worst law in the world" (other countries are considering or already have stricter laws), the DMCA is only a stepping stone toward more restrictive laws and more restrictive technologies (CBDTPA, TCPA, Palladium). What the research community can do is to act through established channels to influence the lawmakers and make its case heard: support EFF, write your congressional representatives, participate in ACM and IEEE policy-making. There is also an upcoming conference on law and policy of digital rights management at Berkeley, Feb. 27 – Mar. 1, 2003. The Q&A session focused on two topics: how did the content industry manage to get the DMCA enacted? By using a catchy slogan – "piracy must stop" – and lots of lobbying $$$. The second question was what can the computer industry and academia do? Rally behind a strong clear theme and lobby policy makers.

Special Evening Panel on Palladium

Lucky Green, Cypherpunks; Peter Biddle, Microsoft; Seth Schoen, EFF

Summarized by Seung Yi

First, Peter Biddle provided a brief overview of Microsoft's approach for the trusted computing project named Palladium. Palladium is an architecture to protect software from other software (even Windows :) and provide a trusted computing platform. Palladium is a security architecture that will be deployed with newer versions of Windows running on machines with tamper-proof hardware components as described in TCPA. Based on this trusted component or Secure Computing Platform (SCP), as Microsoft names it, authenticated booting procedure and

SCP acts as the core of a security architecture that even the machine's owner cannot bypass. By relying on SCP and other trusted software components built on top of SCP, there are certain parts of the operating system that can be trusted by third parties, and with this capability Microsoft claims to be providing trusted computing. More details on Palladium can be found in an article by Seth Schoen at *http://www.activewin.com/articles/2002/pd.shtml*. Also, Microsoft has a Q&A on Palladium available at *http://www.microsoft.com/presspass/features/2002/jul02/07-01palladium.asp*.

Lucky Green was our second speaker. He used his slides to present the concern he had with the proposed TCPA/Palladium architectures. Basically, his points are:

1. TCPA/Palladium is driven by the vendors to make the PC the core of home entertainment by providing a tamper-proof support for digital rights management (DRM), although it is carefully marketed as the solution for trusted computing.

2. TCPA/Palladium can be used to stifle competition that does not have such support. Green gave an example of Windows vs. Linux today. Even though a user can install Linux on a system, there are certain things that can't be done unless the user also installs Windows. By the same logic, it will be still possible to use a TCPA-equipped PC without installing Palladium OS or other similar operating systems, but the user will not be able to access digital music, digital movies, or even her/his own Word file protected by TCPA. Green pointed out a couple of potential abuses of such systems, not surprisingly things not mentioned in the Palladium specification. By invalidating access to Word documents, for example, the vendor can force the users to buy a newer, accessible version of Word. An OS vendor may be able to block certain "undesirable" applications from running on any user's machines. Green's slides are available at the Cypherpunks Web site at *http://www.cypherpunks.to*.

Seth Schoen maintained a somewhat neutral position between Peter Biddle and Lucky Green, pointing out the potential benefits of the proposed architectures and some concerns.

One of the biggest concerns expressed by members of the audience was the possibility of Palladium being used as a DRM platform or, even more alarming, the base platform to implement a 21st-century Big Brother capability. There were also a couple of questions on what part of these proposed architectures is actually new. Most of the concepts proposed in the architectures were already proposed and implemented a couple of decades ago in trusted computing base efforts like KSOS.

For those who wish to learn more about the issue, Ross Anderson provides a nice FAQ on TCPA/Palladium at *http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html*.

Steven Levy wrote an article on the issue in *MSNBC/Newsweek*, which is available at *http://cryptome.org/palladium-sl.htm*.

Panelists also pointed the audience to the discussions on two mailing lists: *cryptography@wasabisystems.com* and *cypherpunks@lne.com*. Archives of these two mailing lists are available at *http://www.mail-archive.com/cryptography@wasabisystems.com/* and *http://www.inet-one.com/cypherpunks/*.

Refereed Papers

OS SECURITY

Summarized by Prem Uppuluri

Security in Plan 9

Russ Cox, MIT; Eric Grosse, Rob Pike, Sean Quinlan, Bell Labs; Dave Presotto, Avaya Labs

This won the Best Paper award. The chair of the session noted, interestingly, that the three authors who were at Lucent when the paper was published are still at Lucent.

Russ Cox emphasized that the main contribution of this paper is a simple security architecture built on a small trusted code base that is easy to verify, understand, and use. The security architecture was developed for the Plan 9 operating system of Lucent Bell Labs.

The authors believe that the main security concern in a system is not the protocols or the algorithms. Instead, buggy servers, confusing software, and poor configurations are usually responsible. Hence, the emphasis of the paper is on the design of a simple security architecture, rather than the algorithms and protocols used, though they have been described for concreteness.

The main component of their architecture is an agent called factotum (derived from the proverbial servant who has the power to act on his master's behalf and has all the keys to the master's possessions). Factotum is built on the same idea as an SSH agent – each user has a factotum process that is responsible for the user's keys. A factotum effectively takes over responsibilities such as authentication and security interactions with other processes. It thus "frees" other software from dealing with these issues. Cryptographic code is no longer compiled with programs but is handled by the factotum, thus allowing for easy updates to crypto software.

An important security consideration is the storage of the secure keys. Factotum stores the keys in the volatile memory, and so the keys need to be backed up. Storing the key encrypted on a shared file system is possible as long as the keys are not the authentication keys. Encrypting the keys with a user password is also not a good solution. since an attacker can use a dictionary attack to break the key. Hence, the authors describe secstore, which is a file server for encrypted data. secstore is based on an encrypted key exchange called PAK.

The paper also describes other security issues, such as protecting factotum from debuggers.

Despite its advantages, there were a few problems. A person from Mitre Corporation asked whether choosing a poor password made factotum susceptible to a dictionary attack. The speaker acknowledged that it did. Another issue, raised by Whitfield Diffie from Sun Microsystems, was whether the architecture could be easily added to UNIX. The authors conceded that it is difficult to add to the existing operating systems but presented an argument that the ideas behind the architecture described can be used in other OSes.

Linux Security Modules: General Security Support for the Linux Kernel

Chris Wright and Crispin Cowan, WireX; Stephen Smalley, NAI Labs; James Morris, Intercode; Greg Kroah-Hartman, IBM Linux Technology Center

LSMs were designed to compensate for the poor security provided by the Linux kernel, which is the same as the classical UNIX security model, in which root is all-powerful. The main goal of the project is to create a security module API that has low overhead (acceptable to Linus, whom Chris Wright called the "dictator"), is minimally invasive, and satisfies the disparate needs of many security projects.

LSM started in April 2001 and involves over 550 people. It basically provides a framework to implement access control models as pluggable kernel modules.

The main design issues that were considered in the design of LSM included: (1) interposing at a level deeper than system-call level, (2) providing a thin mediation layer called hooks that is agnostic with respect to the security model, (3) making LSM restrictive by allowing a module to either allow or deny an access, and (4) allowing module stacking.

In justifying these design decisions, the authors pointed out that system calls, while a natural choice for inter-positioning, are inefficient and may lead to race problems. Hence, they decided to go deeper into the kernel. In particular, LSM provides an interface that allows modules to interact with internal kernel objects. LSM allows a subject to perform a kernel operation on an internal object by placing hooks in the kernel code just ahead of the access to a resource through the system call. LSM is restrictive in its hooks in that a security module intercepting the hooks can either allow the access or deny it. In order to keep the design simple and minimally invasive, the LSM project is limited to supporting core access control functions required by the current security projects. Sometimes security policies need to be composed. The design of LSM forces the decision on how to compose policies on the modules.

The rest of the paper describes the implementation of LSMs. Finally, the speaker concluded that LSM is efficient, producing about 0–2% overhead in micro-benchmarks and 0–0.3% in macro-benchmarks. Currently, LSM is being merged into Kernel 2.5 and the interface is being refined as pieces are submitted to Torvalds. The work is available at *http://lsm.immunix.org*.

There were questions in the audience as to whether any sanity checks were performed for the modules. The speaker said that code reviews and verification of modules were being done by others.

Using CQUAL for Static Analysis of Authorization Hook Placement

Xiaolan Zhang, Antony Edwards, Trent Jaeger, IBM T.J. Watson Research Center

Xiaolan Zhang discussed the use of a static analysis tool, CQUAL, in verifying LSM authorization hook placement. This work revealed potential vulnerabilities in LSM.

Xiaolan first gave a description of a vulnerability in the security hook security_ops->file_ops->llseek(file) as a convincing reason for the need to verify.

She then described the aim of the work, which was to verify the following two problems: complete mediation and complete authorization. For the former, verification involves checking that whenever a user tries to control a resource, some LSM authorization hook mediates. The latter involves verifying that the set of requirements necessary for prior mediation in the authorization process are met in all the paths to the operation that seeks to control the object.

In case of complete mediation, the authors label the resource to be accessed as a controlled object and the operation accessing the resource as a controlled operation. In order to verify that an LSM authorization hook is executed on a controlled object, before it is used they first identify the controlled objects as, for example, files, inodes, superblocks, tasks, or modules. They then use static analysis to associate the authorized object with those used in the controlled operation. In the next step, they identify all possible paths to the controlled operation. They use typical C semantics. All inter-procedural paths are defined by call graphs, and among these paths they identify those that are needed for analysis.

The authors use CQUAL, a type-based static analysis tool that helps find bugs in C programs. As a first step, the authors annotate the data structures in the program with one of two types: unchecked and checked. In particular, all the controlled objects are initialized to the type unchecked, while all function pointers used in a controlled operation are marked as checked. Authorizations

upgrade the object's type to checked. Since the source code is large, annotation by hand was not feasible. Hence the authors extend GCC and use a set of Perl scripts to annotate the code automatically. Type errors indicate possible vulnerabilities.

Using the above techniques they were able to find a couple of exploitable CQUAL type errors. They also had a large number of false positives.

Asked whether there could be other vulnerabilities that may have been missed, the speaker replied that they had some confidence in the result since the approach was generic and wasn't designed to find any one particular error. Another question was on whether the flow insensitivity of CQUAL was a deterrent. The speaker replied that flow insensitivity only increases false positives and does not result in false negatives. The last question was how the work handled function pointers. This was done by manually annotating function pointers in headers. CQUAL can detect function pointers that have been assigned to some variables.

INTRUSION DETECTION/ PROTECTION

Summarized by Haining Wang

Using Text Categorization Techniques for Intrusion Detection

Yihua Liao and V. Rao Vemuri, University of California, Davis

Yihua Liao presented a new approach to modeling program behavior in intrusion detection by using text categorization techniques; this approach eliminates the need to build program behavior databases or learn individual program profiles.

In his talk, he briefly described text categorization, in which text documents are grouped into predefined categories based on their content, and its usage in information retrieval. The vector space model is used to transform documents into vectors. A word-by-document matrix A is used for a collection of documents, where each entry represents the occurrence of a word in a document and can be computed in several different ways – weighting, frequency (f) weighting, and term frequency–inverse document frequency (tf-idf) weighting. They used as a machine-learning method the k-Nearest Neighbor (kNN) classifier, which calculates the similarity between an unknown document and training samples and looks at the class labels of k-nearest neighbors to predict the class of the unknown document.

To profile a program behavior in a much more general and efficient way, the authors treated each system call as a "word" and the set of system calls generated by the process as the "document." Each process is converted to a vector, and the intrusion detection becomes text categorization. Based on the kNN classifier, the program behavior is classified into different categories, which determines normal or intrusive. The advantages include limited system-call vocabulary so that no dimension reduction techniques are needed; use of simple binary categorization; and, as mentioned above, no individual program profiles to learn.

The experiments for testing the kNN classifier were conducted over a 1998 DARPA BSM data set, which provided a large sample of network-based attacks embedded in normal background traffic. The performance of kNN classifier with the tf-idf weighting technique was measured by the Receiver Operating Characteristic (ROC) curve that plots intrusion detection accuracy against false positive probability. The results show that the k=10 is a better choice than other values for achieving a faster detection rate. Also, they compared the tf-idf with f weighting techniques. Although f weighting achieved a higher initial detection rate, tf-idf weighting reached the 100% detection rate much faster. To detect attacks more effectively, the kNN anomaly detection can be easily integrated with signature verification.

Detecting Manipulated Remote Call Streams

Jonathon T. Giffin, Somesh Jha, Barton P. Miller, University of Wisconsin, Madison

Jon Giffin's talk covered how to detect destructive system calls issued by remote execution systems such as Condor and Globus. The detection was based on the pre-execution static analysis of the binary program, in which specifications were automatically generated. A model representing all possible remote call streams that the process could generate was built. As the process executes remotely, the local machine builds optimizations into the model incrementally, ensuring that any call received remains within the model.

The model is a finite-state machine – either a non-deterministic finite-state automaton (NFA) or a push-down automaton (PDA). The construction of the automaton is accomplished in three stages: by (1) deriving the control flow graph (CFG) from each procedure in the binary program; (2) converting the collections of CFGs into a collection of local automata; (3) composing these local automata at points of function calls internal to the application, and then generating the interprocedural automaton that models the application as the whole.

Two metrics determine the usefulness of the model: precision and efficiency. To improve precision, null-call insertion and call-site renaming techniques are employed. To improve efficiency, stack abstractions and null-call insertion are used. During their prototype implementation, they observed that PDA is more precise than NFA because it provides context sensitivity. However, PDA has a state explosion problem – a stack may grow to be unbounded, leading to high overhead. To solve this problem, the

maximum size of the runtime stack is bounded.

Finally, Jon summarized his talk by highlighting the important ideas of the paper: (1) specifications are generated automatically from binary code analysis; (2) a finite-state machine is built that models correct execution; (3) the push-down automaton (PDA) is precise but suffers high overhead; (4) a bounded PDA stack and null calls make the use of a precise PDA model possible.

Type-Assisted Dynamic Buffer Overflow Detection

Kyung-suk Lhee and Steve J. Chapin, Syracuse University

Kyung-suk Lhee gave an introduction to buffer overflow attacks, especially the well-known stack-smashing attack: the return address of a function is overwritten so that the malicious code is injected into the stack, and so the control flow is directed to the malicious code when the function returns. The key idea of the proposed scheme is that a table in the executable file is built at compile time since the size of the buffer can be known, and the sizes of buffers are checked with the table at runtime.

Kyung-suk presented an overview of their implementation, in which they: (1) built the "type table" that holds types (sizes) of automatic and static variables; (2) maintained heap variables in a separate table by intercepting malloc(); and (3) looked up the "type table" to check buffer size using wrapper functions for the vulnerable copy functions in the C library. The prototype was implemented by extending the GNU C compiler on Linux. Each object file was augmented with type information, leaving the source code intact. To delay making the "type table" until runtime, each object file was given a constructor function "ctor" to build the type table. The range checking was done by a function in a shared library.

Their implementation is transparent since source files are unmodified, and programs are compiled normally using the supplied makefile in the source distribution. Type table–appended object files are compatible with native object files. Protected buffers cannot be overflowed or exploited. Moreover, compared with other approaches, this one is harder to bypass and faster than comprehensive range-checking techniques.

The limitations of the scheme include the following: (1) there are two cases where they cannot determine the size of automatic buffer: alloca(), or allocated buffer, and variable-length arrays; (2) the scheme is unable to determine the type of function-scope variables; (3) it is vulnerable to attacks that do not depend on the protected C library functions; and (4) it cannot protect the parameters of the function that defines a nested (function-scope) function. (The fourth point was not mentioned in the paper.)

ACCESS CONTROL

Summarized by Michael Hohmuth

A General and Flexible Access-Control System for the Web

Lujo Bauer, Michael A. Schneider, and Edward W. Felten, Princeton University

Lujo Bauer presented a new access-control system for Web services. He said that there are already many access-control systems that protect an increasing amount of private data, such as photos or medical records. The problem with existing solutions is that many implement only a simple, fixed application-specific policy, and because of that it is hard to express more complex policies or to get these mechanisms to interoperate.

The authors suggest a new, flexible, and general solution that is application- and policy-independent based on proof-carrying authorization (PCA). In this system, clients submit to the Web server all facts that are required to prove that the client is allowed to access a Web page, formulated in higher-order logic. In addition, the client submits a proof of the propositions that are needed before it can access the server. This moves the (generally undecidable) problem of proving the propositions from the server to the client. The server only needs to check the proof (which is decidable), and the client can construct the proof using application-specific, decidable logic.

In their implementation, the authors modified a standard Web server using applets for generating propositions and for checking client-submitted proofs. On the client side, they use an HTTP proxy that hides all server transactions from the standard Web browser. This proxy handles proof challenges from the server by trying to construct proofs for them. If it is missing facts required for constructing the proof, they ask fact servers (which are specialized Web servers). Bauer said the proxy could be integrated into the browser as a plug-in, but they wanted it to be as browser-independent as possible.

Bauer presented performance results for their system. As the performance is bound by the number of transactions between clients, fact servers, and Web servers, the system uses caching and speculative proving to avoid unnecessary transactions. Clients cache protected URLs and facts and try to guess and speculatively prove the server's challenges before the server actually generates them. Servers cache proven propositions and client-generated lemmas. As a result, the performance overhead of the system is promising.

Bauer concluded the talk with the statement that formal tools and methods have a place in the real world.

Jonathan Shapiro (Johns Hopkins University) asked how one would deal with

revocation of facts in the light of caching. Bauer answered that facts can have a timeout by including a reference to the current time.

Another audience member asked whether submitting endless unfinished proofs to the server would be a potential DoS attack on the system. Bauer affirmed but said that a similar attack existed with previous systems, and now that the server does not have to prove access propositions itself, it had, in a sense, "less to do" than previously. Another question was whether access policies have to be stored in the server. Bauer answered that was convenient but not required.

Access and Integrity Control in a Public-Access, High-Assurance Configuration Management System

Jonathan S. Shapiro and John Vanderburgh, Johns Hopkins University

Jonathan Shapiro presented OpenCM, a new configuration management system designed to support high-assurance development in open source projects.

Shapiro started his talk with the question: what is configuration management (CM)? He proposed two different answers that he deemed too limiting (it keeps track of versions of files or collections of files) before he presented his answer: a CM system should keep track of "lattices of DAGs of attributed BLOBs" (i.e., relationships between file-version trees) and bindings from file versions to names in a workspace, together with file metadata.

The authors started developing a new CM system because they needed support for developing an operating system (EROS) that can be certified by the highest of the Common Criteria assurance levels, EAL7 (comparable to the former orange-book level A1). This assurance level requires software development to be traceable, auditable, reproducible, and access-controlled, and it also requires high data integrity. As EROS is developed as an open source project, additional requirements were that the CM system needs to support many contributors, but not all of them should have write access to the main repository. Aside from the fact that no existing CM system supports all of these requirements, Shapiro also mentioned the need for a CM system that "actually worked" and that existing commercial offerings did not support the open source development model very well.

OpenCM is designed to protect against such threats as modifications (of the source code repository) by unauthorized users, modifications from compromised clients, compromises through the underlying operating system, impersonation of a source repository, and falsification of repository content. OpenCM reaches these goals by establishing a chain of integrity and authorization for each change request, and by using transactions to commit changes to the repository.

Shapiro explained that the key idea for meeting the integrity requirement was to realize that most of the objects a CM system stores (such as file contents of a particular revision) never change (because of its archival character); he referred to these objects as frozen objects. Therefore, the cryptographic hash of a frozen object's contents also never changes and can be used as a name to reference the frozen object. Whenever such a name is de-referenced, the contents of the object can immediately be checked for integrity. The integrity of mutable objects is ensured by cryptographic signatures.

A transacted change to the repository is reduced to the addition of new data as frozen objects and the atomic revision of a single mutable object, the branch to which the change is committed. Access to mutables is controlled using access-control lists.

Shapiro then identified a number of possible weaknesses of OpenCM: content compromise using a stolen repository-server key, history disclosure using exposed hash names of previous versions, and separate evolution of database and client-server protocol schemas. He proposed solutions or recovery possibilities for each of these problems.

Shapiro concluded his talk with a demo of OpenCM running on his laptop.

Petros Maniatis (Stanford) asked whether more than one server can be authoritative for a given repository. Shapiro answered that OpenCM does not support this mode of operation, as distributed updates to a single repository would be unfeasibly complex. However, changes can be committed to a (nonauthoritative) replicated repository and merged into the authoritative repository later.

An audience member asked whether changes should be signed. Shapiro replied that they shouldn't, but that the subject would be too complex to discuss as part of his talk. He suggested taking the issue offline.

Richard Wash (CITI Michigan) asked what would happen if two nonidentical frozen objects happened to have the same content hash. Shapiro said that a hash collision would be noticed but could not be recovered from. He said that such a collision would be extremely unlikely, though.

HACKS/ATTACKS

Summarized by George M. Jones

Deanonymizing Users of the SafeWeb Anonymizing Service

David Martin, Boston University; Andrew Schulman, Software Litigation Consultant

This paper presented an analysis of the SafeWeb anonymous Web browsing service. The anonymizing service was halted in November 2001.

The goal of the service was "to help oppressed international users" who wanted to view Web content that their country/organization/ISP/etc. prohibited. It also had appeal to corporate and home users. Requirements appear to have been speed, ease-of-use, unmodified content, and no client-side modifications or settings.

The main method employed was to disguise the connection so that all browsing was proxied through HTTPS connections to SafeWeb.com. Both URL and contents were encrypted. Possible attacks were presented. Some involved sending content (JavaScript) that induced the browser to go directly to the source Web site. SafeWeb's rewrites were not perfect.

Some conclusions: SafeWeb took the wrong default stance by blocking known bad (e.g., java-script) elements and allowing all else. Its use openly defied local policies/laws.

VeriSign CZAG: Privacy Leak in X.509 Certificates

Scott G. Renfro, Yahoo!

Scott Renfro examined VeriSign's CZAG extension as an example of embedding sensitive information into X.509 certificates. He then considered the general case of sharing certified information with multiple parties.

In 1997 VeriSign asked end users to (optionally) include country, zip, age, and gender (CZAG) information when registering for class one certificates. Users assumed that this information would be kept private and only shared with trusted parties. But there were problems. It was protected only by weak encryption (XOR), there was no revocation enforcement, it was available in a public LDAP directory, indexed by email, and easy to crawl.

Next, Renfro listed goals, design constraints, and possible alternate implementations for allowing certificate authorities to share sensitive information without unnecessary disclosure. The "most obvious" solution is to keep sensitive information in a centralized database which responsible parties can query with their own credentials. Another option would be to use better key management and stronger encryption for sensitive information. A third set of options involves various methods of putting control of sensitive information in the user's hands, departing completely from the X.509 certificate approach.

How to Own the Internet in Your Spare Time

Stuart Staniford, Silicon Defense; Vern Paxson, ICSI Center for Internet Research; Nicholas Weaver, University of California at Berkeley

Paxson gave very plausible visions of Internet attacks to come based on recent experiences with Code Red and Nimda and made the case for the creation of a "cyber Center for Disease Control (CDC)."

"What could you do if you owned a million hosts?" Launch DDoS attacks, wipe out disks, rummage through email and credit card databases, crack passwords, send "trusted" messages, stage cyberwarfare between nations or acts of outright terrorism.

"How do you own a million hosts?" Short answer: worms. The Morris Worm owned 10% of the Internet. Code Red (2001) peaked at an infection rate of 1900 infections/minute. Monitoring of two class B networks showed 300,000 infected hosts. The larger the vulnerable population [read: IIS install base], the faster it spreads. Nimda spread itself several ways, including by looking for back doors installed by Code Red. "These viruses form an ecosystem."

"We couldn't resist designing better worms," Paxson said and then outlined several methods future worms could use to spread quickly by intelligently splitting up scans of the IP address space. Peer-to-peer networks and "contagion" worms present other fruitful methods of spreading malicious code. "If you have the entire hit list [vulnerable hosts] and infected a few and divide up the list, then it is possible to infect 1M-10M hosts in seconds. These time-scales are way beyond human response."

So what's the answer? A "cyber CDC" that would identify outbreaks, coordinate response, do rapid analysis, help resist infection, watch traffic, set strategic direction, and foster research. "This may sound hard, but what's the alternative?"

Q: What are you proposing beyond CERT/FIRST?

A: Automated response, instant analysis.

Q: How seriously do you take the threat of embedding viruses in pictures and other file types?

A: Nonexecutable files are probably not a significant worry.

Q: Do we have a need for more centralized analysis of worms?

A: This is very ripe for research. Open community analysis has been very helpful . . . but we still don't know what Nimda does.

Q: Can you comment on the use of worms to patch security holes?

A: That seems like a non-starter. There is a very large liability issue.

SANDBOXING

Summarized by Prem Uppuluri

Setuid Demystified

Hao Chen, David Wagner, University of California at Berkeley; Drew Dean, SRI International

Hao Chen addressed a critical problem with the use of UID-changing calls, asserting that setuid and seteuid suffer from many flaws. They are poorly

designed, lack proper documentation, are widely misunderstood and, hence, misused by programmers. As an example he pointed out that a system-call setuid(0) (setuid to root) shows different behavior in Linux and BSD. In Linux it sets only the UID to 0, whereas in FreeBSD it may set all the three UIDs – SUID, UID, and EUID – to 0. Another problem he illustrated was that sometimes the UID-changing calls may not actually succeed. For instance, the system-call seteuid(geteuid()) seems like an identity function and so is expected to succeed, but may not necessarily do so.

To address such problems, the authors studied the kernel sources for these calls and then compared the precise semantics of the calls across Linux, Solaris, and FreeBSD. They did this by constructing a formal model of user IDs as a finite-state automaton (FSA). This FSA helped them find some of the pitfalls of the UID-changing calls and also helped them identify the semantic differences of these calls across the three operating systems.

The authors describe the model-extraction algorithm which constructs an FSA. The states of the FSA contain the values of UID, SUID, and EUID. A transition is labeled with one of the UID-changing calls. From each state there is one transition labeled with each UID-changing call. Each transition leads to a state which contains the values of the three UIDs after the execution of the UID-changing call associated with the transition.

Using the finite automaton they built, they were able to verify a number of inconsistencies: a man page of RH Linux 7.2 fails to mention setuid capability and a man page of setreuid in FreeBSD 4.4 mentions incorrectly that unprivileged users may change real UID to effective UID. They were also able to identify that the implementations of the calls across the operating systems were different.

At the end of the paper, they provide guidelines to the proper use of these system calls. For instance, they suggest that setesuid be used where available as it has very explicit and clear semantics and sets the three user IDs independently. They also suggested that users check for errors in the return code of system calls. In particular, a good technique to confidently drop privileges is to first drop the privilege permanently, try to regain the privilege, and ensure that the program cannot regain the privileges. Further information on their work is at *http://www.cs.berkeley.edu/~hchen/research/setuid/*.

Secure Execution via Program Shepherding

Vladimir Kiriansky, Derek Bruening, Saman Amarasinghe, MIT

Saman Amarasinghe argued that it is not possible to attain zero bugs in code. Thus it is necessary to look at other techniques to prevent the bugs from being exploited. The key point on which they base their work is that one who owns the program counter controls the code. An attacker who is prevented from hijacking the program counter may overwrite data but cannot control the code. Based on this observation, they described their approach, which they call program shepherding.

In program shepherding, all control-flow transfers during a program execution are monitored, and security policies are defined to determine allowable transfers. Program shepherding can be done in two main ways. One way is to instrument application and library code prior to execution and to add security checks around every branch instantiation. They argue, however, that this approach is not viable or applicable. The approach they took was to use an interpreter.

The naïve approach to interpreting, however, is very slow. Hence they used a dynamic optimizer (DynamoRIO base system built in association with HP labs and MIT) in order to improve the performance of the interpreter. In addition they ensured that non-control flow instructions did not get interpreted. They further reduced overhead using indirect branch lookups.

To measure the effectiveness of their approach, they used a set of vulnerable applications: stunnel, groff, ssh, and sudo. They were able to foil all exploits, with no false positives. Their performance numbers were also very good, with the overhead around 8% due to their interpreter.

Someone asked how this approach differed from fault isolation techniques; Saman replied that in this approach the isolation is at a lower level of granularity.

A Flexible Containment Mechanism for Executing Untrusted Code

David S. Peterson, Matt Bishop, and Raju Pandey, University of California at Davis

David Peterson described a variety of sandboxing techniques and explained the design of their framework, which draws from these different techniques.

Peterson started by describing the different design alternatives available for sandbox creations. In particular he addressed:

1. Representation and organization of privileges in the sandbox. They first identified resources that needed to be protected, including device components, file systems, network components, and signal components. When a sandbox is created, one or more of the components are attached to it. Initially only the sandbox creator is given privileges for these components, but privileges to other processes in the sandbox can be added.

2. Location of enforcement mechanisms. The authors described the various choices to insert the enforcement mechanisms: runtime environment, sandboxed program, user space, and OS

kernel. They chose the OS kernel, as it allowed them to use the system-call API.

3. Passive or active monitoring. Passive monitoring involves changing the system-call execution such that any enforcement mechanisms are checked before the system call is allowed to proceed. This involves modification of the system call. Active monitoring requires that an external process monitors the program. Both these techniques have advantages: active monitoring is flexible, and passive monitoring introduces low overhead. The authors decided to use a mechanism that allows for either or both of the monitoring techniques.

4. Whether to group sandboxes globally or locally.

5. Whether the access control mechanisms must be mandatory or discretionary. Their design provides both options.

6. How to guard access to sandbox-related objects.

Peterson discussed many other options and described the design of their sandbox. The overhead introduced by their system varied from 0.3 to 4.0%.

An audience member wondered whether they were considering making their system into an LSM module. The reply was an affirmative.

WEB SECURITY

Summarized by Haining Wang

SSLACC: A Clustered SSL Accelerator

Eric Rescorla, RTFM; Adam Cain, Nokia; Brian Korver, Xythos Software

SSL is much more CPU intensive than ordinary TCP communication, because of the cryptographic computation, especially the RSA operation in the SSL handshake. To offload the cryptographic overhead, an accelerating proxy is introduced. However, the accelerator becomes a single point of failure, and multiple accelerators are only a partial solution since any connection on a failed accelerator is lost. The real problem with SSL is that the user does not know whether the transaction is complete and so is unwilling to re-submit the transaction.

Eric presented a better approach, a clustered SSL accelerator, in which all nodes in the cluster share the connection state. When any node fails, the remaining nodes are able to take over all connections that terminated on that node with no interruption in service. Failures are invisible to the end user; this process is called active session failover. The design principles of SSLACC are embodied in the three laws of clustering: (1) "all nodes must generate the same data," and all nodes behave as one virtual device; (2) "cluster then commit," which requires tight control of the TCP stack; and (3) it is safe to transmit unclustered data if you can reproduce it.

Note that they do not cluster data but use a clustered TCP relay. Data is automatically buffered by the client. Only full records can be processed at the server, however, and sometimes records are bigger than the TCP window size (especially during slow-start). The proposed solution is to ACK a partial record: cluster the record data read so far and ACK the partial read. To keep cluster updates as small as possible, only a minimal amount of state is transmitted so that the other nodes can reproduce the original state on failover. In conclusion, the most desirable properties in a clustered accelerator are scalability, high availability, and the ability to run on cost-effective hardware.

Infranet: Circumventing Web Censorship and Surveillance

Nick Feamster, Magdalena Balazinska, Greg Harfst, Hari Balakrishnan, and David Karger, MIT

This paper won the Best Student Paper award. Nick Feamster presented Infranet, a way to circumvent Web censorship and surveillance that consists of requesters and responders communicating over a covert tunnel. The key idea is that the Web browser requests the censored content via Infranet requester as a local proxy, which in turn sends a message to an Infranet responder. The responder retrieves this content from the appropriate origin Web server and returns it to the requester, then the requester forwards the received content to the browser. The covert communication tunnel securely hides the exchange of censored content in normal, innocuous Web transactions.

Then he described what kind of censors people might want to get around, which include restrictive government, corporate firewall, etc. Basically, there are two classes of attacks mounted by the censor: discover attack, where the censor monitors the Web traffic for unusual-looking access attempts and traffic; and disruptive attack, which blocks communication between endpoints by preventing access to certain Web sites or attempting to block access to circumvention software. Related systems – e.g., Triangle Boy, Peekabooty – and their vulnerabilities were mentioned.

The design goals of Infranet include: (1) deniability for clients – the censor cannot confirm that any client is intentionally downloading information via Infranet; (2) statistical deniability for clients – the browsing patterns are indistinguishable from innocent clients; (3) covertness for servers – the censor cannot discover a server that is serving censored content and so cannot easily block such a server; (4) communication robustness – the Infranet channel should be robust in the presence of censorship activities designed to disrupt request/transfer of censored content; and (5) reasonable performance.

In the downstream communication, censored data is embedded in images

and recovered later by shared secret. However, steganography is not ideal, because it cannot reuse a cover image. Web cams, where images are constantly changing, would be a better choice. In the upstream communication (i.e., requesting), the requester divides the hidden message into multiple fragments, each of which is translated to a visible HTTP request by a modulation function. The mapping function was a design trade-off between covertness and bandwidth consumption. The reasonable performance is achieved by taking advantage of the asymmetric bandwidth requirements of Web transactions, which require significantly less upstream bandwidth than downstream bandwidth.

Trusted Paths for Browsers

Zishuang (Eileen) Ye, Sean Smith, Dartmouth College

Eileen Ye first pointed out that the human user is the true client, not the machine; however, the communication between the Web browser and the user is a neglected component of the server-client channel. Simply ensuring that the machine draws the correct conclusion does not suffice if the adversary can craft material that nevertheless fools the human. According to their definition, Web spoofing is malicious action causing the reality of the browsing session to be significantly different from the mental model a reasonably sophisticated user has of that session.

They tried to reproduce Princeton's Web spoofing experimental work done in 1996, but they did not succeed, due to the advances in Web technology and browsers' user interface. So they conducted their own experiments to demonstrate the weak link between the human user and the Web browser. To foil Web spoofing, a trusted path was created between the browser and its human user. Through this trusted path, the browser can communicate relevant trust signals that the human can easily distinguish from the adversary's attempts at spoof and illusion.

Besides clearly communicating with the security-related information, the attributes of the trusted path should include: inclusiveness (working on all interfaces), effectiveness (expressing the security information in a way the user can easily understand), minimal intrusiveness, and minimal user activity. To meet these requirements, a colored boundary approach was taken, known as synchronized random dynamic (SRD) boundaries. In an SRD environment, all windows have colored boundaries. A blue boundary window (containing server materials) indicates an untrusted window, while an orange boundary window (containing browser materials) indicates a trusted window. The window boundary has two styles: inset and outset. At random intervals, the browser would change the styles on all its windows. The random pattern of the boundary style cannot be predicated by the server, so the server cannot forge a window image to impersonate the real window.

Mozilla was chosen as the base browser for implementing SRD. There are three steps to implement SRD: (1) add special boundaries to all browser windows; (2) make the boundaries change dynamically; and (3) make all windows change synchronously. To resolve the address-blocking problem (i.e., an SSL warning window blocking other windows), a reference window running in a separate process was introduced. The reference window changes its image by random number to indicate the boundary style. In usability studies, three test scenarios were included: (1) without reference window, (2) a full SRD approach, and (3) a CMW-style approach. The conclusions drawn from a user study were: it works! See the paper for additional suggestions.

GENERATING KEYS AND TIMESTAMPS

Summarized by Michael Hohmuth

Toward Speech-Generated Cryptographic Keys on Resource-Constrained Devices

Fabian Monrose, Qi Li, Daniel P. Lopresti, and Chilin Shih, Bell Labs, Lucent Technologies; Michael Reiter, Carnegie Mellon University

Michael Reiter presented this talk on what he said was fairly speculative research: the extraction of a key usable for cryptographic purposes from a biometric such as voice. The main criteria for a usable system would be that it works reliably and efficiently even with constrained resources such as cell phones, PDAs, and other wearable devices and that key extraction should be difficult even if an attacker gets access to the samples of the biometric.

In this research, the authors concentrated only on voice, since that is the natural interface for many wearables. Also, voice is a dynamic biometric in that the user can change a "passphrase" by speaking a different phrase or changing intonation, and thus can have many different keys. Reiter stated clearly that he indeed meant voice, not the phrase recognized and recovered from voice; the latter would have many fewer features and would mean a loss of information and thus key length when compared to pure voice.

Reiter first presented an overview of their system. It works by taking a voice sample, generating a list of small segments through digital signal processing,

extracting from the segments a vector of binary features (which Reiter called feature descriptors), and, using each feature, selecting a key element from a two-columned key table. As not each repetition of the passphrase yields exactly the same feature descriptor, the algorithm also needs to reconstruct the correct feature descriptor by searching within a given Hamming distance of the extracted feature descriptor (key reconstruction).

Reiter said that he and his colleagues have presented parts of this system earlier in other publications (IEEE S&P 2001, ACM CCS 1999); in this talk, he would focus on the implementation, on the signal-processing part, and on empirical analysis of the strength of generated keys.

The authors first implemented their system on the Yopi, a Linux PDA powered by a 206MHz StrongARM CPU. This implementation suffered from a low-quality microphone built into the device and a poor OSS sound-driver implementation. In a second implementation, the authors switched to the iPAQ 3600, also equipped with a 206MHz StrongARM.

As an illustration of the harsh realities developers face when using resource-constrained devices such as these, Reiter explained that silence elimination was an important step in their signal-processing step and showed waveforms of recorded "silence" generated by these devices. Instead of silence, the Yopi recorded static. The iPAQ's waveform was distorted by the device's automatic gain control.

Using these devices, key reconstruction currently works practically with a Hamming distance of up to five features (on future systems, the authors expect to be able to support six features). Based on typical Hamming distances when comparing the feature descriptor originally recorded and a capture of the passphrase spoken by the real speaker, this limits the number of distinguishing features that can be supported on these platforms to about 30. Using the best-known attack, an adversary that can only randomly guess features needs 2^40 multiplications to recover the key.

The authors also looked at other attacks on the signal-processing part that they deemed more promising than random guessing: another person uttering the same passphrase, and recovery of the original passphrase using the original speaker's voice by way of sophisticated text-to-speech synthesis and diphone cut-and-paste from a huge database of phrases spoken by the original speaker. Reiter mentioned that an attacker does not need a database as large as theirs; 20 minutes of good-quality recordings of the speaker would contain enough phonemes to synthesize 50 percent of the passwords they tried.

Interestingly, these impersonation attacks did not yield better results than random guessing. Reiter said he and his team had expected that these attacks would break their system, and they were surprised that they did not. It is unclear why these attacks do not work. Reiter speculated that he and his coauthors did not carry out the attacks correctly, or that speech synthesis is too immature, but he said that this kind of attack must be expected to become more powerful in the future.

In conclusion, Reiter said that the feasibility of using voice for generating strong keys is still unproven, but their results indicate that the approach is promising and can be implemented.

Paul van Oorschot (Cloakware) asked about the security that can be expected if an attacker obtains a recording of the speaker speaking the passphrase. Reiter replied that the authors would make no claims about that case.

Neil Daswani (Stanford University) asked whether their cut-and-paste attacks included cases in which whole subphrases of the passphrase were concatenated. Reiter said that this type of attack was included in the study.

Another audience member asked whether they tried speech synthesis using AT&T's Natural Voices product, released about one year ago, and how it compares to other speech-synthesis products. Reiter said that he does not know of AT&T's product and hence cannot compare it.

Secure History Preservation Through Timeline Entanglement

Petros Maniatis and Mary Baker, Stanford University

Petros Maniatis started out by referencing Jonathan Shapiro's talk earlier in the conference. He said that Shapiro was concerned with preserving history of a collection of files; his work has the same goals, but in a broader context, that of preserving the sequence of a host of events in a large distributed system.

Maniatis said that in this work, history is defined to be the temporal ordering of system events such as storing a file on a disk or signing a document. Such events can occur in unrelated, distributed components. However, there are circumstances in which the order of two such events is important even if they did not occur in the same system, for instance when referencing prior art in patent disputes.

The speaker went on by giving a more elaborate motivating example in which an investor, Marti, ordered a sell of shares of some company. The next day, something bad happens to the company. Marti's broker sells the shares a day later, just before the stock price plummets prior to the bad news becoming public. Later, the SEC accuses Marti of insider trading, and now Marti would like to prove that he ordered the sell of shares before the bad event occurred. Maniatis insisted that this example was purely fictitious, which amused those audience members who had followed that week's US national news revelations about MCI/Worldcom's creative bookkeeping.

The authors set out to build a system that is designed to preserve the sequence of events "long after the 'historians' leave," under the assumption that no party trusts another. In their approach, each component maintains a local history and a local view of the global history. Components safeguard the integrity of the portions of history they know about and trust only themselves or information that can be proved. Other

requirements on the system were efficiency, scalability, survivability, and aggressive decentralization. To address these requirements, the authors developed a method for "timeweaving," interconnecting local histories with each other so that a global history can be reconstructed.

Maniatis explained that each component's history consisted of a hash chain of commitments of local events. The elements of the chain are called time steps; they contain the current local time, a description of the event, and an authenticator. The authenticator links the time step to the previous one in the timeline using a one-way hash function. Then, precedence can be proven by giving enough information for walking a thus-established hash chain. To avoid having to disclose each and every event between two events of interest, the chain includes special events that reference each other and that form a skip list for jumping over a number of other events.

Timeline entanglement, or timeweaving, works as follows: components regularly publish timeline samples for other components to witness, and witnesses commit published samples in their own timeline. Then witnesses send the originating component an entanglement receipt, which includes a precedency proof stating that all events in the publisher's past occurred before all events in the witnesses' future.

Maniatis then covered implementation aspects. Here, the challenge was to find a balance between storage overhead needed for storing authenticated hash chains and the number of disk accesses and computation steps needed to compute precedence proofs. The authors use a new data structure, RBB-Trees, which bounds the maximal number of disk accesses needed to compute an authenticator to three. Their performance study shows that in a network of 1200 1GHz PCs that generate events every second and in which each pair of hosts entangles every 10 minutes, each PC uses about 8% of its resources.

Matt Blaze (AT&T Labs) asked whether a possible attack on the proposed system would be to add many histories, making entanglement between all of them impractical. Maniatis affirmed, saying that if there was not enough framework to connect two events, no precedence could be proved.

WORK-IN-PROGRESS REPORTS, AKA QUESTIONS FROM PETER HONEYMAN

Summarized by George M. Jones

Session Chair: Kevin Fu

At the work-in-progress (WiPs) session, presenters are given five minutes to talk about current work and take questions. Due to the presentation format and space limits, these summaries are guaranteed to contain omissions, gross inaccuracies, and misrepresentations of presentations on some fine work. You are encouraged to contact the presenters for more complete, less sketchy information. Also see *http://www.usenix.org/events/sec02/wips.html* for the authors' own abstracts.

Preventing Privilege Escalation

Niels Provos, CITI, University of Michigan

Provos presented the idea of separating applications into two parts, privileged and unprivileged, citing the example implementation in OpenSSH, which he claimed had prevented the "gobbles" attacks from taking over CITI.

Memory Accounting Within a Multi-tasking Language System

Dave Price, Rice University

Price talked about a solution to the problem of memory accounting in an environment (Java) where all tasks share a single heap. The solution proposed was to do accounting during garbage collection. This is done by starting at the root of each task and walking the reachable memory tree, charging the first task for shared memory.

Semantics-Aware Transformation and Anonymizing of Network Traces

Ruoming Pang (with Vern Paxson), Princeton University and ICSI Center for Internet Research

This talk presented work on a way to scrub network traces of private information using the BRO IDS. Stream reassembly is done (see work presented by Paxson et al. last year), and users are given the ability to write AWK-like scripts that can tag/scrub their data before it is entered into the trace.

Clilets: Web Applications with Private Client-Side Storage

Robert Fischer, Harvard University

Fischer presented a new system called "clilets" to implement privacy on the Web. The user sends a request to the Web server, the Web server sends a "clilet" to a multi-domain sandbox, the sandbox sends HTML to HTML verifier, HTML verifier sends HTML to Web server, which sends it to client. The server and clilet work together to create the HTML. Peter Honeyman asked, "This sounds like Java VM – what's new?"

Checking Linux Kernel User-Space Pointer Handling with CQUAL

Robert Johnson, and Sailesh Krishnamurthy (with John Kodumal), University of California at Berkeley

Johnson talked about a system called CQUAL that solves the problem of verifying correct uses of user and kernel pointers in the Linux kernel. The C type system does not support this, but CQUAL does. Using this system, an actual bug was found and fixed in the Linux 2.4.19 kernel.

Segmented Deterministic Packet Marking

John-Paul Fryckman, University of California at San Diego

Fryckman proposed a solution for tracing attacks across the Internet. It involves adding "back-pointers" to packets in the IP headers. The first (edge) AS

and every subsequent AS adds its own AS number to the packet. It was claimed that with at most 17 AS numbers, the entire Internet could be covered.

Turing: A Fast Software Stream Cipher

Greg Rose, Qualcomm Australia

Rose presented initial work on a new fast, simple stream cipher called Turing, designed for use in cheap, slow, small CPUs with little memory. It uses keyed non-linear transformation and was inspired by work on "tc24." The net effect: an Athlon can do 3 cycles/byte. "If it works and is secure, it will be the fastest stream cipher in software."

Active Mapping: Resisting NIDS Evasion Without Altering Traffic

Umesh Shankar, University of California at Berkeley

Ways of avoiding IDSes have been known for some time (Ptacek, Newsham, 1998). These problems stem from uncertainty about what packets reach end systems and how they are interpreted. Most of these problem can be overcome by normalizing the traffic and interpreting the TCP stream as the target system would. To do this, the authors built a database of the systems and types of systems on their local net and performed IDS on normalized data as the end system would see it.

Making Software Resistant to DoS Through Defensive Programming

Xiaohu (Tiger) Qie (with Ruoming Pang and Larry Peterson), Princeton University

This talk presented the case for building robust network infrastructure (routers, systems) by applying improved programming techniques and tools. They built a C toolkit, allowing programmers to specify general resource usage policies. It does some flow analysis, performs consistency checks, and uses sensors/actuators. It was used in real software (Linux networking code). Results were mixed.

VFiasco – Toward a Fully Verified Operating-System Kernel

Michael Hohmuth, TU Dresden

Hohmuth and associates believe that "formal methods can be worthwhile," and they deny the conventional wisdom that "OS verification is an intractable problem." With that starting point, he presented their work on Fiasco, a microkernel OS written in a C++ subset and their results in proving one class. To the question of how long it would take to prove the whole OS, Hohmuth answered, Three to four years.

Wormhole Detection in Ad Hoc Networks

Yih-Chun Hu, CMU

Your humble summary writer admits to note-taking failure for this talk and kindly asks that you visit the author's Web site:
http://monarch.cs.rice.edu/papers.html

A Snapshot of Global Internet Worm Activity

Dug Song, Arbor Networks

Song presented work on monitoring Internet worm activity by monitoring large chunks of unused Internet address space. The work is unique in that for 1/N SYNs to port 80, they reply with an ACK and then log payloads. Using this method they can track attacks individually and can see DDoS and backscatter traffic. Song also presented data on the rise, continued prevalence, and interactions of Code Red and Nimda.

Off-the-Record Communication

Nikita Borisov, University of California at Berkeley

In online conversations as in the real world, you may want conversations to be private, but you may want repudiation...the ability to deny that you said something. PGP and friends use long-lived keys that provide non-repudiation. This is not good for casual conversation. The author then presents work on a protocol for instant messaging to solve this problem. It involves frequent key renegotiations, symmetric authentication, revealing the MAC key in the clear, and introduction of delays.

Plutus – Enabling Secure Sharing of Persistent Data

Erik Riedel, Seagate Research

Riedel presented file system work done at HP to address the problems of both sharing and protecting data, dealing with key management, and distributing the encryption workload. Their system pushes key management and encryption to the edge, uses untrusted servers that only do verified writes, supports keys for groups of files, not users, and is client centered. It is built on AFS using secure RPC.

A Signature Matching Engine for BRO

Robin Sommer, TU Munich, ICIR

Sommer said that traditional signature matching just compares signatures to net traffic, whereas BRO reuses existing signatures and uses regular expressions. BRO supports bi-directional signatures and uses knowledge about target (this is Apache server; IIS exploit does not matter).

Honeyd: A Virtual Honeypot Daemon

Niels Provos, CITI, University of Michigan

Provos presented his work on "honeyd," which implements a small, low-interaction virtual honeypot. It can simulate arbitrary TCP services, listen on up to 65,000 IPs at one time. It reads the nmap fingerprint database and can respond appropriately to impersonate anything in nmap DB. It can simulate arbitrary virtual routing topologies, lie to traceroute, and simulate packet loss and various services. You can proxy attackers back to themselves.

Peter Honeyman asked, "This is not part of your research. How do you ever expect to get your Ph.D. [from me] working on stuff like this?"

NordU USENIX 2003

NordU2003 – The fifth NordU/USENIX Conference February 10–14, 2003 Aros Congress Center Västerås, Sweden

Announcement

We would like to welcome everyone to Västerås (100 km east of Stockholm) and to the NordU2003 – The fifth NordU/USENIX Conference.

As UNIX and Free Software becomes more and more widely adopted in corporations and academia we see a need to stay ontop of current trends.

The NordU/USENIX Conference offers a venue for developers, administrators and users of UNIX and UNIX like operating systems, to meet, talk and learn.

Jon "maddog" Hall will deliver one of the keynotes. Mr Hall has a long history of being involved in our community. We will take a special look at how Open Source Software, such as GNU/Linux, *BSD, Open Office and GNOME, is being used in government. For developers we have some exciting invited presentations on Java Technology as well as GNU Compiler Collection family of tools

We look forward to meeting you all in Västerås!

Programme

Monday – Wednesday
February 10–12
Tutorial Programme

Thursday – Friday
February 13–14
Technical Programme
Papers
Exhibition
Sponsors Presentations

Topics that will be covered

Security
Operating Systems
Desktop
Tools
Backup Solutions
Applications
Open Source/Free UNIX
Interoperability

.SE EurOpen DKUUG EurOpen NUUG SEUUG FUUG NordU2003

http://www.nordu.org/NordU2003/

Announcement and Call for Papers **USENIX**

12th USENIX Security Symposium

http://www.usenix.org/events/sec03

August 4–8, 2003 **Marriott Wardman Park Hotel, Washington, D.C**

Important Dates for Refereed Papers

Paper submissions due: *January 27, 2003*
Author notification: *March 20, 2003*
Camera-ready final papers due: *May 13, 2003*

Symposium Organizers

Program Chair
Vern Paxson, *ICSI*
Program Committee
Steve Bellovin, *AT&T Labs—Research*
Dan Boneh, *Stanford University*
Crispin Cowan, *WireX*
Drew Dean, *SRI International*
Kevin Fu, *MIT*
Peter Gutmann, *University of Auckland, New Zealand*
Richard Kemmerer, *University of California, Santa Barbara*
Patrick McDaniel, *AT&T Labs—Research*
John McHugh, *CERT® Coordination Center*
Radia Perlman, *Sun Microsystems*
Niels Provos, *University of Michigan*
Dawn Song, *Carnegie Mellon University*
David Wagner, *University of California, Berkeley*
Dan S. Wallach, *Rice University*
Elizabeth Zwicky, *Greatcircle Associates*
Invited Talks Coordinator
Matt Blaze, *AT&T Labs—Research*

Symposium Overview

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

If you are working on any practical aspects of security or applications of cryptography, the program committee would like to encourage you to submit a paper. Submissions are due on January 27, 2003.

This symposium will last for five days. Two days of tutorials will be followed by 2.5 days of technical sessions, including refereed papers, invited talks, Work-in-Progress reports, panel discussions, and Birds-of-a-Feather sessions.

Symposium Topics

Refereed paper submissions are being solicited in all areas relating to systems and network security, including:

- Adaptive security and system management
- Analysis of malicious code
- Analysis of network and security protocols
- Applications of cryptographic techniques
- Attacks against networks and machines
- Authentication and authorization of users, systems, and applications
- Automated tools for source code analysis
- Denial-of-service attacks
- File and filesystem security
- Firewall technologies
- Intrusion detection
- Privacy preserving (and compromising) systems
- Public key infrastructure
- Rights management and copyright protection
- Security in heterogeneous and large-scale environments
- Security of agents and mobile code
- Security of Internet voting systems
- Techniques for developing secure systems
- World Wide Web security

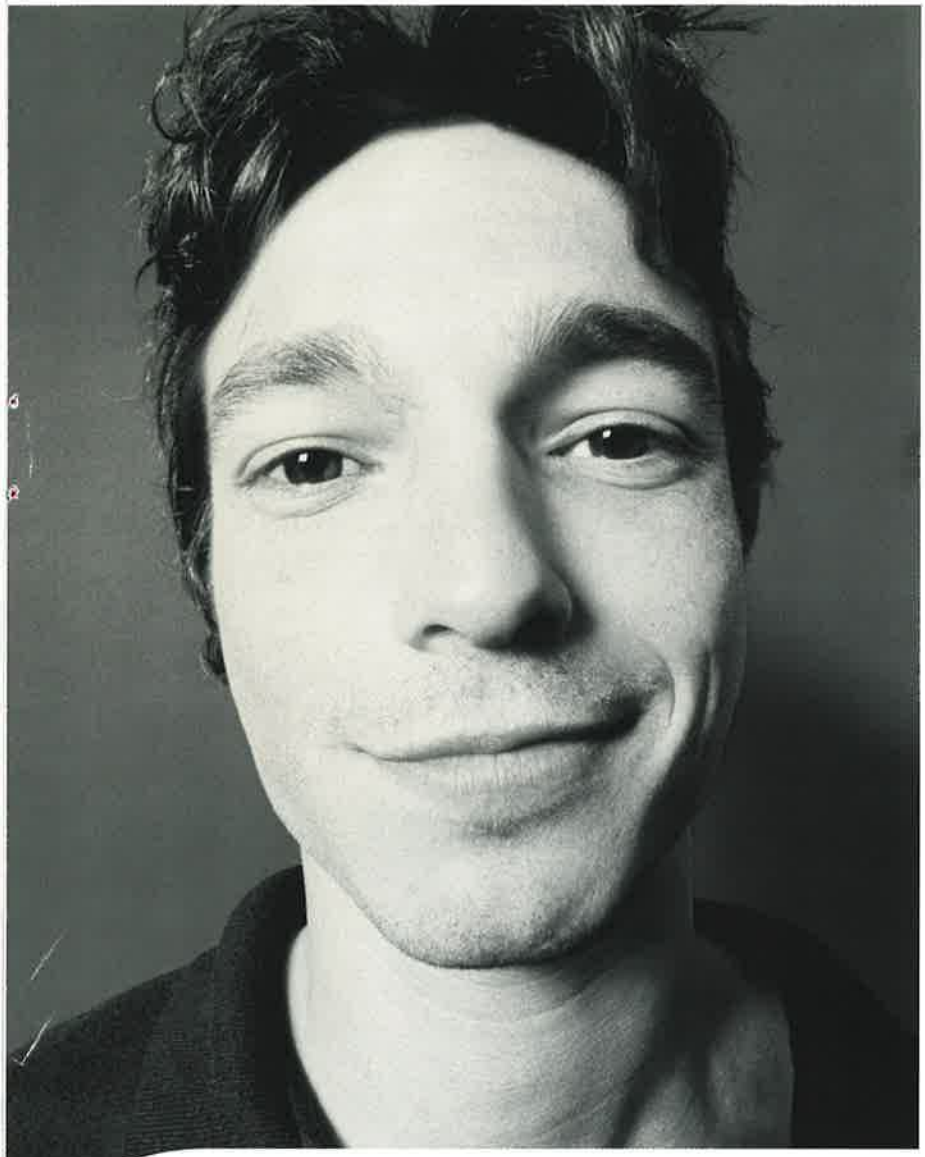
Since USENIX Security is primarily a systems security conference, papers regarding new cryptographic algorithms or protocols, or electronic commerce primitives, are in general discouraged.

Refereed Papers (August 6–8)

Papers that have been formally reviewed and accepted will be presented during the symposium and published in the symposium proceedings. The proceedings will be distributed to attendees and, following the conference, will be available online to USENIX members and for purchase.

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.



"That's *Mr.* SmartyPants to you."

Be the first to know what's happening across your Web systems with SiteScope.®

When it comes to the status of your Web systems, being a "Know-It-All" is a good thing. In fact, it's a critical part of your job. SiteScope from Freshwater Software lets you proactively monitor your network services, system resources, and applications. You're alerted to availability issues and other problems before they can rage out of control. Best of all, **it takes just 20 minutes to install, configure and begin using SiteScope**—no matter how many servers you're monitoring. Looks like people will just have to get used to that look of supreme confidence on your face.

See how easy it can be to proactively monitor your Web systems:

Download a FREE trial of SiteScope today at www.freshwater.com/KnowIt2

Freshwater Software®
a Mercury Interactive company

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

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

WEB SITES

http://www.usenix.org

http://www.sage.org

EMAIL

login@usenix.org

COMMENTS? SUGGESTIONS?

send email to ah@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.

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).

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

;login:

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

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

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

**********3-DIGIT 208