ZD JOURNALS

June 1999 • $9.00
Vol. 5 No. 6
www.zdjournals.com/sun

Inside Solaris™
Tips & Techniques for users of Sun Solaris

Midnight Commander

by Alan Orndorff

A long time ago, on another computing platform, Peter Norton Computing released Norton Commander. This became my favorite file management program. As I wandered further and further into the UNIX realm, I found it hard to believe that a program like this wasn't available on UNIX. Finally, I came across Midnight Commander, shown in **Figure A**. It offers more features than Norton Commander and, unlike Norton Commander, it runs on a variety of different computing platforms.

Acquiring the software

If you point your browser to **www.gnome.org/mc**, you'll find the source code and binary executables for the program. The precompiled binaries generally lag behind the source code releases, though. Compiling Midnight Commander couldn't be easier. If you're using gcc, simply run Configure followed by Make. In a few minutes, you'll need to type *make install* to place it into the usual /usr/local directory hierarchy. Also, as usual, if you aren't placing GNU binaries into /usr/local, this can be easily modified.

For a complete listing of all compile time options, consult the online documentation found on the Web site mentioned earlier. There are some GUI options that you can try, but these are beta and you may or may not be able to get them to work on your system.

The basics

`mc` starts the shell. Once you've started the program, you'll see that the screen of Midnight Commander is divided into essentially three main parts.

There are two panels laid side by side, which contain directory information and a command line. In addition to the three main areas, there's also a menu bar across the top and a list of function keys across the bottom of the screen.

Each panel can display the same or different directories on your hard disk. Most file operations will take place in the panel with the selection bar. Some commands use the other panel as the default for file operations such as copy or move. You can also issue commands directly from the command line.

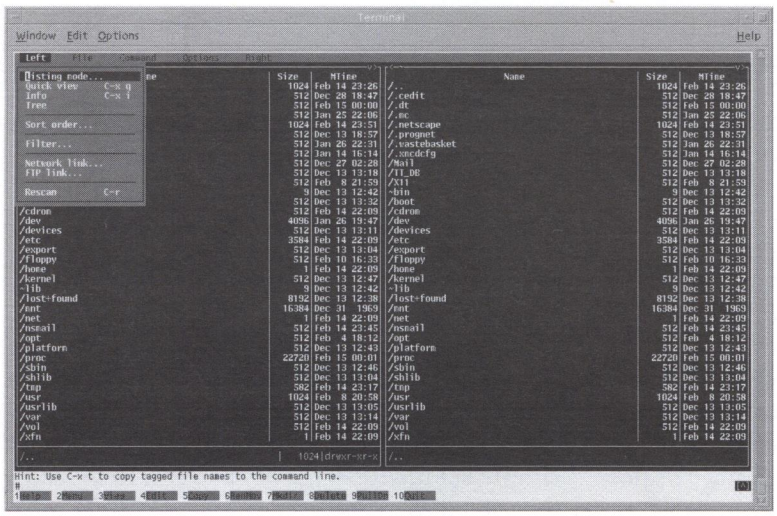


Figure A: *Here's the interface for Midnight Commander.*

In this issue:

ZIFF-DAVIS a SOFTBANK company

Like most programs, Midnight Commander supports the use of hotkeys. The hotkeys that the program uses are an approximation of the GNU Emacs editor keys. For a complete list of all the hotkeys, consult the Midnight Commander manual.

Midnight Commander's main use is file management. You have access to a wild assortment of customizations to get the job done according to your liking.

Listing modes

Listing modes define how the file structure is displayed inside each panel. To change listing modes, press [F9], and then select either the right or left pulldown menu to change the listing mode for the right or left panel. Then, select one of the four modes: Full, Brief, Long, or User.

The *Full* directory view shows the file name, the file size, and the last modification time. The *Brief* view shows only the filenames, and uses two columns in each panel.

The *Long* view shows the permissions, the number of links, the owner, the group, the size, the last modification time, and the filename. The Long view mode will use only one panel that will cover the length of the window and the other panel will be unavailable.

If the default listing modes aren't what you're looking for, you can use *User* mode. When you choose this option, you choose which fields to view and the display format to view them with. The display format must start with a panel size; this can be either half (two panels) or full (one panel). After the panel size, you may specify a two-column mode similar to the listing mode that Brief uses. After this, you add the name of the fields you wish to display.

The following are your field choices: filename, file size, an alternate form of the file size, file type, last modification time, last access time, file creation time, permission bits, the permissions in octal format, the number of links to the file, the numeric GID, the numeric UID, the owner of the file, the group of the file, or the inode of the file.

You'll also need to specify how the fields are separated. You can choose to use spaces, dots, or a vertical line to denote where one field ends and another starts. Another option is to have the program display an asterisk (*) if the file is tagged or a space if it isn't.

Modes

If you don't want to view two directories side by side, you don't have to. You could set one of the panels to one of three different modes instead. When you select a file in the one panel, the other panel will display information about the file, depending on which mode you have it set to. These modes are Info, Tree, and View.

- **Info**—Information, similar to ls -l, about the currently selected file and if possible the current file system.
- **Tree**—A simple tree view of the file system. This displays directories and sub-directories only, and you can view the files in the other panel. You can also change directories by selecting a directory and pressing [Enter].
- **Quick View**—When you select a file in the other panel, the panel that's set up to Quick View will display the contents of the selected file, if possible. This works best with text files.

Sorting and filtering

As you've probably guessed, you can sort the file listing based on a number of different options: by name, by extension, by modification time, by access time, by inode modification time, by size, by inode, and unsorted. By default, the program will place all of the directories at the top of the listing followed by the files. You can change that if you press [F9], and then select Options | Configuration | Mix All Files. This will mix the files with the directories and sort the output based on the sort action that you've chosen. And lastly, you can sort in reverse order.

Suppose you have a directory of a thousand files or more. You could move the selection bar up and down until you eventually get to the file or files you want to work on, or you could just filter the file listing by name. To do so, simply enter the filter criteria into the filter box, and Midnight Commander will display only those files.

You can also pull a small trick using the filter option. A side benefit of filtering is that the program will display the sum total of all files selected at the bottom of the panel. You can simply set the filter to All Files and then read the total disk space used at the bottom of the panel. This may be easier than using `du`.

Finding files

Sometimes you may need help finding a file, and Midnight Commander can help make this task easier. Press [F9], select Command | Find File, and then enter the starting directory, the name of the file you're looking for, and, optionally, the text string in the file that you're looking for. Then, let the program find it for you. Once you find your file, you can have the program change directories to the one your files are in, or you can panelize the results.

When you panelize the results, one of the panels changes to show the results of the Find command. The panel will display only those files that matched your Find criteria.

If the reason you're looking to find a file is to view its contents or to edit it, this couldn't be simpler. The program has both an internal viewer and internal editor. If you can't remember the exact directory name to start in, simply choose the Tree option from the menu and use the tree to select the starting directory.

Virtual file system support

If you're using Midnight Commander on a UNIX system, you'll have access to three or four virtual file systems. The file systems that Midnight Commander supports internally are FTP, TAR, Network File System, and on Linux—the Undelete File System, if you have it enabled on your system.

FTP virtual file system

Midnight Commander can be set up to access an ftp site similar to a Web browser. You connect to a remote site by simply changing directories to the site you wish to access.

For example, `cd ftp://ftp.netscape.com` will connect you to Netscape's ftp server, where you could ftp the latest version of Communicator. The full syntax for the ftp command is ftp://[!]user:password@remote host[:port][directory].

The bang character [!], user name, password, port, and directory are all optional. The bang character tells the program to use a proxy host when attempting to connect to a remote machine via ftp. Instead of using a bang character, you could modify the mc.ini file and add ftpfs_always_use_proxy to set this for all sessions. If you choose to use the proxy setting, you should also create a file called mc.no_proxy to define local machines that don't need to be accessed via a proxy server. A Web browser will use what's called passive mode to access an ftp site, as opposed to the standard UNIX ftp command, which will use active mode. If you add ftpfs_use_passive_connections to mc.ini, the program will access an ftp site as a passive connection.

Once you're connected to the ftp host, the remote host's file listings will appear in one panel, and you can set the other panel to either your file system or possibly another directory on yet another remote host. Then you'd use the normal Midnight Commander functions to manipulate the files on either system. To disconnect from the remote host, simply `cd` to a directory on your local machine.

TAR virtual file system

Midnight commander supports a read-only TAR virtual file system. When you move the selection bar onto a TAR, gzipped, or compressed file, and press [F3] or [Enter], the program will show a directory listing of all files contained in the TAR archive.

Network file system

Midnight Commander has the ability to support its own Network File System, which is different than NFS developed by Sun Microsystems. To use Midnight Commander's Network File System, the host machine needs to run a special server program called mcserv. You'd then connect to the host machine with Midnight Commander running on a client machine. This gives functionality similar to NFS. Consult the documentation for more information.

Undelete virtual file system

On Linux systems with the Undelete File System enabled, Midnight Commander can act as a front-end to the undelete functionality of that file system. Consult the documentation for more information.

Nifty features

Midnight commander has many neat features. This includes background processing, hotlists, and size reporting.

Background jobs

You can process the File | Copy and Move commands in the background, if you desire. You can also select options to stop, restart, and kill a background job from the Background Job menu item.

Directory hotlist

The directory hotlist is a neat little feature of this program. With it, you can set up a list of directories you normally go to, and after activating the directory list hotkey, select a directory, and you'll be magically transported there. Almost as good as setting up a link to another directory.

Show directory sizes

This can be a very useful feature. Let's say you're running out of disk space and you need to know whose home directory is consuming the lion's share of the shared disk. Simply use Midnight Commander to view the home directories, and then select Show Directory Sizes from the menu.

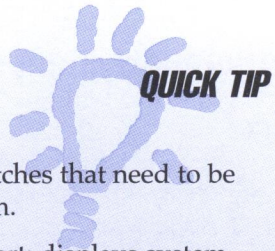
If you're on a large file system, this may take some time. The program will add up all the space used by each directory and their respective sub-directories, and then show the directory names and the total disk space used.

Summary

I've left out some of the other options that Midnight Commander can perform, such as command history or directory compare, as these should be self-explanatory. You should consult the documentation that comes with the program to see all of its uses.

This program seems to be in a constant state of change, which is good, as the authors add even more features to it. This can be a valuable tool in your day-to-day administration duties. Enjoy. ZDJ

SunScan 2000



by Werner Klauser

Sun Microsystems offers a free, downloadable Year 2000 product compliance audit tool. Found at **www.sun.com/year2000/sunscan/**, it requires Solaris 2.3 or higher. SunScan 2000 simplifies Year 2000 preparation by determining the Year 2000 compliance of Sun hardware platforms, operating systems, and other unbundled products listed on Sun's compliant product list. For each Sun system, this tool will generate a report listing all detected products by name and will then make recommendations on how to achieve Year 2000 compliance.

The following information is gathered and generated by SunScan 2000:

- Date the information was gathered.
- Report on system information (hostname, hostid, OS release, kernel and application architecture, hardware provider, domain, and kernel version).
- System resident patches are analyzed against a list of minimum revision Year 2000 list of patches and the following three lists are generated:
  - Lists the Year 2000 patches installed on the system that are Year 2000 Compliant.
  - Lists the Year 2000 patches installed on the system that require an up-revision.
  - Lists the Year 2000 patches that need to be installed on the system.
- Hardware information report: displays system type and prom revision. Any required system type or PROM revision upgrades will be reported in this section.
- Product compliance report: will analyze all installed system packages and map them to a Sun product. After product detection, SunScan 2000 will verify the product's Year 2000 compliance status. If the product is determined not to be Year 2000 compliant, a message indicating the upgrade path (if known) will be displayed.
- Packages not mapped to a Sun product will be listed under the title "Unresolved packages." Therefore, this list may or may not contain packages that belong to third party software, hardware, and device drivers.

To keep current with changes in Year 2000 compliance status, updated versions of SunScan 2000 will be emailed to the address you provided during registration.

Using Sun's patches and checking it with its patchdiag and SunScan 2000 tools allows you to keep up with the necessary patches to keep Sun's Solaris operating system up and fully alive in the transition to the year 2000.

Open file systems with Solaris

by Paul A. Watters

In the past, local area networking was dominated by proprietary, single-platform data exchange protocols that were incompatible with the heterogeneous nature of IT development in many organizations. One of the first attempts to inter-connect single-user workstations with large servers was the Networked File System (NFS) protocol used by Solaris to exchange files with PCs by using the PC-NFS software. However, with the development and widespread acceptance of the Internet Protocol (IP) as a means of exchanging data across heterogeneous, wide area networks, a need has arisen for a freely available common networked client/server file system protocol, to improve accessibility to local data.

In this article, we examine the development of the Server Message Block (SMB) protocol for server-side sharing of files and input/output devices, such as printers and parallel ports. Also known as the Common Internet File System (CIFS) protocol, SMB centrally places Solaris systems as reliable servers to many different SMB clients, using the Samba software suite.

Centralization

Many large and small organizations have a requirement for sharing information that's centrally stored on a server. Centralization of storage and processing time was a key strategy in the design of many early information systems, where mainframe or mini-computers received requests from dedicated client terminals, processed them, and returned the desired response. However, as the speed and storage capacity of terminals has increased to the point where they're often considered powerful workstations in their own right, the impetus for centralizing hardware capacity has decreased.

However, from an information systems point of view, it's often necessary and desirable to have a logical server for responding to client requests for information that's unique and that's required for later retrieval. For example, the Oracle database server allows databases to be physically distributed across many different operating systems and servers, while guaranteeing reliable and logically transparent access to unique records held on different servers.

In a physical sense, the database is decentralized, but it's logically consistent in responding to queries from clients, regardless of physical origin. This is especially the case with Internet-based front ends to database clients, where access might originate from anywhere in the world.

Standardizing data exchange

To ensure that this kind of data sharing requirement is adequately met in heterogeneous client/server systems, it's necessary for disks and other physical devices to be shared. The development of the Internet Protocol (IP) is one step in the direction of standardizing data exchange between computers. However, while the File Transfer Protocol (FTP) is useful for exchanging files with clients halfway across the world, it's very inconvenient for repetitive data exchange in local area networks, where a new session must be established for each transfer.

Many companies have sought to fill this gap by providing proprietary and often single-platform data exchange solutions for local area networks, so that physical devices on other computers can be locally mapped as logical devices. Novell's NetWare, for example, facilitated the sharing of remote printers and other devices such as network drives for PCs.

Alternatively, Microsoft and Intel developed the Open Net File Sharing Protocol, which was intended to be a platform-independent way of sharing disks and other networked devices. Commands would be issued using Server Message Blocks (SMB), over several types of protocols (TCP/IP, NetBEUI and IPX/SPX, to name a few), in a simple request-response format. Access to resources is determined at the share level, with a single password required to access a specific device, or at the user level, where a client user name and password must be supplied.

Windows-based systems have SMB clients built into their File and Printer Sharing drivers. However, it's on the server side of things that Solaris has come to play a central role in the way that SMB has been deployed in large

organizations, through the development and implementation of the Samba software suite developed by Andrew Tridgell.

Samba

Figure A shows how Samba, running on Solaris, can be used to grant client access to printers and hard disks (perhaps containing centralized databases). A request is issued by a client using a message block transmitted using TCP/IP, NetBIOS, or IPX/SPX, which the server then responds to, after authentication has been performed against a user's access rights (or against the shared resource's access list).

Under Solaris, Samba runs as a daemon process, which automatically reloads its configuration every minute from the smb.conf file. This file defines access levels and authorization requirements for accessing any devices on the server. To allow access to a shared disk called Oracle, we can include a section like this:

```
[Oracle]
        path = /Oracle
        writeable = false
```

This would give read-only access to the logical partition /Oracle. To allow printing rights to any client using the guest account on the SPARC printer, we could include a section like this:

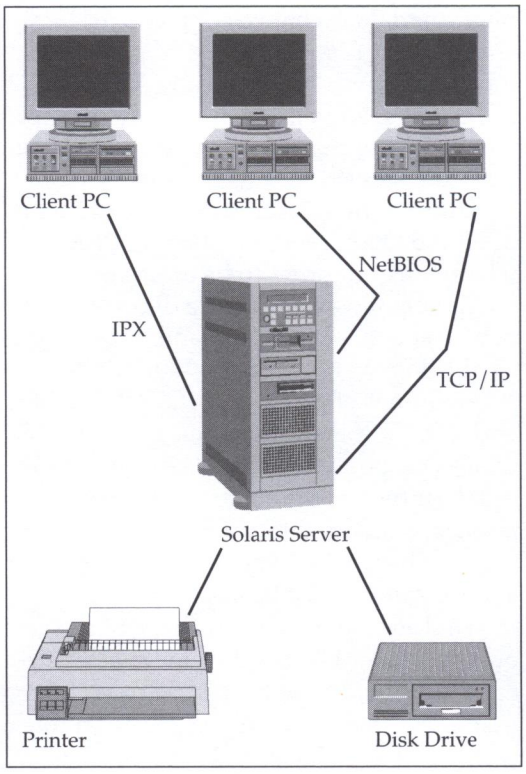


Figure A: *Use Samba to integrate different resources on your network.*

```
[SPARCprinter]
        path = /var/spool/SPARC
        read only = true
        printable = true
        guest ok = true
```

User names from client networking software can often be mapped to match those of the server, thereby automating the login process.

In a secure environment, passwords can also be encrypted, and an SSL layer activated for all data transfers. This should lay to rest any security concerns that administrators might have about exchanging sensitive data as plain text through networks. In addition to the server software for Solaris, there's also a client called smbclient that can be used to access resources from both UNIX and non-UNIX servers, although many UNIX to UNIX connections will still be made through NFS.

In summary, Samba and the SMB protocol will further consolidate the central role of Solaris systems as servers for client PCs, and as data warehouses for database systems that require reliable data storage and processing.

Samba 2.0

According to the press release issued at

au1.samba.org/samba/whatsnew/samba2.0.press.html

the latest version of Samba (Samba 2.0) has now been identified as the world's fastest Windows server, using the Ziff-Davis NetBench ® benchmarking suite. The major innovations in this release include a full implementation of the Windows NT Domain authentication protocols (including Primary Domain Controller), with the ability to add NT clients to the domain.

In addition, a Web-based, fully integrated Samba Web Administration Tool (SWAT) is included, facilitating client-based server administration. Some bug fixes for Solaris are also contained in the new version, including appropriate loop termination of the Samba daemon during password changing, and large file support.

Installation

Installation of Samba 2.0 is more complicated than previous versions, due to the number of decisions that must be made about how shared resources are administered and how

services are run (as stand-alone daemons, or through the inetd Internet daemon). After the sources have been downloaded from samba.org, they should be unpacked into a temporary directory ready for compilation. The first task, though, is to read the Fine manual—in this case, the Readme file, and the UNIX_INSTALL.TXT file in the documentation subdirectory. These contain most (but not all) of the steps required to successfully compile and install Samba 2.0 on Solaris.

Makefiles are created using a standard GNU-style configure script, which is executed in the source subdirectory by

```
./configure
```

To obtain a list of options that can be passed to the configure script, like changing the default installation directory from /usr/local, use the command

```
./configure --help
```

To compile the sources, type

```
make
```

On Solaris 2.6, the sources compile without any errors, but with a few inevitable but non-serious warnings such as:

```
Compiling lib/system.c
lib/system.c: In function `sys_readdir':
lib/system.c:304: warning: return from
➥incompatible pointer type
```

These can be safely ignored. The binaries can be installed under /usr/local/samba (default) with the command

```
make install
```

This also installs the SWAT files in the default hierarchy. Now that the binaries have been installed, the Samba documentation wisely suggests making a cup of coffee before configuration, as it can be tricky. This advice should be heeded, as there are several points at which configuration can go astray. We'll deal with them now.

Configuration issues

An smb.conf file can now be created for testing purposes. Its location is defined in the original Makefile, and is generally found in the /usr/local/samba/lib directory (although some administrators might choose /etc for consistency). Here's a test of the samb.conf file:

```
workgroup = saturn

[homes]
    guest ok = no
    read only = no
```

This configuration should allow users on the host machine to access their home directories with an appropriate client, after locating the server in the Network Neighborhood in NT.

Before making the daemon live, it's wise to test the syntax of smb.conf, as well as the server installation, using the testparm program. When we run this program on the sample smb.conf file, the following appears:

```
# /usr/local/samba/bin/testparm
Load smb config files from
➥/usr/local/samba/lib/smb.conf
Processing section "[homes]"
Loaded services file OK.
ERROR: lock directory
➥/usr/local/samba/var/locks does not exist
```

Obviously, the make install hasn't created all of the appropriate directories for us, so we must do it ourselves with:

```
# mkdir /usr/local/samba/var/locks
```

When we run testparm again, the problem is solved, and the configuration has been loaded successfully as indicated by the output:

```
#/usr/local/samba/bin/testparm
Load smb config files from
➥/usr/local/samba/lib/smb.conf
Processing section "[homes]"
Loaded services file OK.
Press enter to see a dump of your service
➥definitions
```

The service definitions show values for many global parameters such as the workgroup, netbios name and aliases, the server identity string, DNS resolution, password encryption, and guest access (which incidentally was disabled in the test smb.conf file above). They're very useful to examine

when debugging an installation. Most values can be set explicitly within smb.conf.

Now that our services look okay, we need to decide whether to run the daemons smbd and nmbd as individual daemons or from inetd. For security-conscious sites, it might be preferable to use inetd as little as possible, and start services from /etc/rc3.d. Running as a daemon also means that the response time will be faster than inetd services (an important consideration if our server has many clients).

To start the Samba daemons, and to ensure that they're automatically restarted after booting, create a file called */etc/rc3.d/S99sambaserver*, and make a soft link to /etc/init.d/sambaserver. This will make Samba one of the last services to start after all the networking and security features have been loaded after booting. S99sambaserver should contain something like that shown in **Listing A**.

Now start the server manually with the command:

```
/etc/init.d/sambaserver start
```

When we check the status of the server with:

```
/usr/local/samba/bin/smbstatus
```

we get the message

```
Samba version 2.0.0
Service      uid       gid       pid
➥machine
Share mode memory usage (bytes):
   1048464(99%) free + 56(0%) used +
   ➥56(0%) overhead = 1048576(100%)
   ➥total
```

Once we've run the startup script, we can check the server status with the command `smbclient- Lmars`. The output of this command is shown in **Listing B**.

Listing A: *Script to automate startup (etc/rc3.d/ s99sambaserver)*

```
killprocess () {
              pid=`/usr/bin/ps -e |
              /usr/bin/grep -w $1 |
              /usr/bin/sed -e 's/^  *//' -e 's/ .*//'`
              [ "$pid" != "" ] && kill $pid
}

case "$1" in
'start')

      /usr/local/samba/bin/smbd -D
      /usr/local/samba/bin/nmbd -D
      ;;

'stop')
      killprocess smbd
      killprocess nmbd
      ;;
*)
      echo "Usage: /etc/init.d/sambaserver { start | stop }"
      ;;
esac

Then make the script executable with:

chmod +x /etc/rc3.d/S99sambaserver
```

Listing B: *Listing the available services for our MARS client with smbclient-L*

```
Added interface ip=1.2.3.4
bcast=1.2.3.255 nmask=255.255.255.0
Password:
Domain=[saturn] OS=[Unix] Server=[Samba 2.0.0]

        Sharename      Type      Comment
        ---------      ----      -------
        homes          Disk
        IPC$           IPC       IPC Service
                                 (Samba 2.0.0)

        Server               Comment
        ---------            -------
        MARS                 Samba 2.0.0

        Workgroup            Master
        ---------            -------
        KCS                  MARS
```

SWAT

The SWAT documentation only contains details for running SWAT within inetd, so let's look at how it's done (we could also have used this technique for installing the Samba daemons as inetd services).

In /etc/services, we need to add a line with a service port identifier like this:

```
swat            901/tcp
```

We also need to tell inetd how to run this new service. So in /etc/inetd.conf, we should add a line like:

```
swat    stream  tcp     nowait.400  root
➥/usr/local/samba/bin/swat swat
```

We need to find the process running inetd, and issue a hangup signal to tell the process to reread inetd.conf:

```
# ps -eaf | grep inetd
   root   156     1  0 12:07:37 ?
   ➥0:00 /usr/sbin/inetd -s
   root  7972
# kill -1 156
```

We should now be able to connect to http://localhost:901 as root and administer the Samba server. However, we should keep in mind that SWAT will rewrite your smb.conf file. If you're uncomfortable allowing SWAT to do this as root on your machine, edit the smb.conf file yourself, or try another GUI tool from the many listed at

http://samba.anu.edu.au/samba/GUI/

Availability

You can find further information regarding Samba from the Samba Web site at

www.samba.org

This site has links to several complete mirrors in different countries and regions. The source is available if you prefer to compile everything from scratch. However, Solaris 2.6 binaries can be downloaded directly from

www.sunfreeware.com

Many useful threads regarding the stability of Samba 2.0 and other SMB client-server issues can be found on the USENET forum

comp.protocols.smb.

Using SNMP, part 2

by Don Kuenz

The Simple Network Management Protocol (SNMP) enables you monitor and alter performance characteristics of network devices, such as workstations, PCs, servers, routers, hubs, gateways, and printers. In "Introducing SNMP, part 1," last month, we described the architecture of SNMP. In this second installment of our two-part series, we'll show you how to install and use a free software package, which implements the protocol. Let's quickly review SNMP before plunging ahead with implementation details.

Review

The SNMP protocol runs at the OSI application layer to provide access to network management information. SNMP provides for two types of networked devices: *managed devices* and *Network Management Servers (NMS)*. A device may simultaneously function as both a managed device and an NMS.

An NMS allows you to read information about managed devices, such as machine architecture and average bytes transmitted. You can also dynamically alter parameters by writing information to a network device. For instance, you can change the system date/time on a managed device or tell an active process to stop running.

Each managed device stores its network information in an object store known as a

Listing A: *An excerpt showing part of the snmpd.conf file*

```
# For specific usage information, see the
# man/snmpd.conf.5 manual page as well as the
# local/passtest script used in the above example.

# ----------------------------------------------

#
# System contact information
#

# It is also possible to set the sysContact
# and sysLocation system variables through the
# snmpd.conf file:

syslocation Right here, right now.
syscontact Me <me@somewhere.org>
```

Management Information Block (MIB). SNMP requires each MIB to contain mandatory, standard objects such as operating system and physical location. It also allows vendors to add optional objects.

Obtaining and installing UCD SNMP

You can peruse the UCD SNMP home page at **www.ece.ucdavis.edu/ucd-snmp** and download the software package at **ftp://ftp.ece.ucdavis.edu/pub/snmp**. This site contains pre-compiled binaries for SPARC Solaris 2.6. If you run any other version of Solaris, you'll need to compile the source code.

You should find it easy to compile the source. Although UCD displays quite a few warnings during the compile, most of the time you can safely ignore all of the warnings. The compile/install procedure generally consists of saving `ucd-snmp-3.5.3.tar.gz` in an appropriate directory, then issuing the following six commands:

```
gzip -d ucd-snmp-3.5.3.tar.gz
tar -xf ucd-snmp-3.5.3.tar
cd ucd-snmp-3.5.3
./configure -prefix=/opt/snmp
make
make install
```

The last command must be invoked as root. Allow us to explain the `./configure` command. UCD SNMP defaults to installing in the /usr/local directory, but Solaris convention dictates using a directory located under /opt. The `-prefix=/opt/snmp` parameter allows us to change UCD SNMP's default directory to /opt/snmp. Or you can forget about convention and leave off the `prefix` parameter all together.

Table A: *NMS applications with a short description*

Application	Function
`snmpbulkwalk`	Queries for a tree of information from a managed device.
`snmpdelta`	Reports changes over time to managed device object(s) with integer values.
`snmpget`	Retrieves information from a managed device's object and sets the current object.
`snmpgetnext`	Retrieves information from the next managed device object, which follows the current object.
`snmpset`	Sets information on a managed device's object.
`snmpstatus`	Retrieves important (IP address, IP packet counts) from the managed device.
`snmptest`	Monitors and manages information on a managed device.
`snmptranslate`	Translates an SNMP object into another form of information.
`snmptrap`	Uses the TRAP request to send information to a network manager.
`snmpwalk`	Retrieves information from all of a managed device's objects, which fall under a specified branch.

The agent

From the first article, you might recall that all managed devices run a process, which is known as an agent. An *agent* provides access to the managed device's MIB object store. It also uses information within the MIB object store to fine tune network processes. The agent should run all of the time. Solaris literature uses the word *daemon* to describe a process that runs all of the time.

SNMP uses a daemon named `snmpd` to add agent functionality to a managed device. `snmpd` requires a configuration file named snmpd.conf. By default, `snmpd` looks for snmpd.conf in the share/snmp directory under your install path. You can also explicitly specify a configuration filename by including the `-c` argument with your invocation. In either case, you must first construct an snmpd.conf file before invoking `snmpd`.

Listing A, on page 9, shows an excerpt from a snmpd.conf file. We happened to set the values of the `system.sysContact.0` and the `system.sysLocation.0` MIB objects from within this file. Keep in mind that there are many other ways of setting an object's value besides using the snmpd.conf file. The compile process makes a file named EXAMPLE.conf, which you can copy to share/snmp/snmpd.conf in order to use it as a starting point for your own configuration.

After you install UCD SNMP and create an snmpd.conf file, you can start the agent by changing to the `sbin` directory under your install path and typing *./snmpd*.

NMS applications

The UCD SNMP software package includes 10 NMS applications. The install step makes these 10 applications when you compile your

source code. **Table A** shows each of the 10 applications along with a short description the application's function.

Listing B shows the general syntax used to invoke an NMS application along with the output produced by the application. Although **Listing B** only shows the `snmpwalk` application, each of the nine other applications uses a similar syntax. The `snmpwalk` application generally takes four arguments. The `-v 2` argument tells `snmpwalk` to use the MIB-II classic protocol. The `hephaestus` argument tells `snmpwalk` that we want to communicate with the SNMP agent on a host named `hephaestus`. We specify a community using the `public` argument. Finally, we use the `system` argument to tell `snmpwalk` the objects we wish to see.

Part 1 explained how the MIB object store arranges objects into a tree-like, hierarchical structure. This article uses the MIB-II standard, which automatically prefixes object names with `.iso.org.dod.internet.mgmt.mib-2` when you invoke an NMS application and leave the leading period off of the object argument. You can also obtain output identical to that shown in **Listing B** with this command:

```
snmpwalk -v 2c hephaestus public
.iso.org.dod.internet.mgmt.mib-2.system
```

In the latter case, which uses an *absolute object name*, the object argument begins with a period, so nothing is automatically prefixed to it.

Let's take a closer look at the output shown in **Listing B**. All of the 36 objects listed belong to the system group. SNMP obtains the values of these objects in various ways. For instance, SNMP obtains the values of `system.sysDescr.0` and `system.sysName.0` by using operating system calls, but it obtains the values of `system.sysContact.0` and `system.sysLocation.0` by looking in the `snmp.conf` file, which we described earlier.

Summary

In this article, we've shown you how to install and use the UCD SNMP software package. SNMP allows you to manage network devices. Management includes interrogating managed network devices, as well as dynamically altering their parameters to change performance characteristics.

The UCD SNMP package contains one agent application and 10 NMS applications. The agent application runs as a daemon named `snmpd`. You can communicate with agents using 10 NMS applications named: `snmpbulkwalk`, `snmpdelta`, `snmpget`, `snmpgetnext`, `snmpset`, `snmpstatus`, `snmptest`, `snmptranslate`, `snmptrap`, and `snmpwalk`.

Listing B: *The snmpwalk NMS application followed by its output*

```
$ snmpwalk -v 2c hephaestus public system
system.sysDescr.0 = SunOS hephaestus 5.5 Generic i86pc
system.sysObjectID.0 = OID:
➥enterprises.ucdavis.ucdSnmpAgent.solaris
system.sysUpTime.0 = Timeticks: (3621792) 10:03:37.92
system.sysContact.0 = Me <me@somewhere.org>
system.sysName.0 = hephaestus
system.sysLocation.0 = Right here, right now.
system.sysServices.0 = 72
system.sysORLastChange.0 = Timeticks: (0) 0:00:00.00
system.sysORTable.sysOREntry.sysORIndex.1 = 1
system.sysORTable.sysOREntry.sysORIndex.2 = 2
system.sysORTable.sysOREntry.sysORIndex.3 = 3
system.sysORTable.sysOREntry.sysORID.1 = OID:
➥.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB
system.sysORTable.sysOREntry.sysORID.2 =
➥OID:.iso.org.dod.internet.snmpV2.snmpModules.snmpM2M
system.sysORTable.sysOREntry.sysORID.3 = OID:
➥.iso.org.dod.internet.snmpV2.snmpModules.snmpVacmMIB
system.sysORTable.sysOREntry.sysORDescr.1 = The Mib module for
➥SNMPv2 entities.
system.sysORTable.sysOREntry.sysORDescr.2 = The Manager-to-Manager
➥MIB module.
system.sysORTable.sysOREntry.sysORDescr.3 = View-based Access
➥Control Model for SNMP.
system.sysORTable.sysOREntry.sysORUpTime.1 = Timeticks: (0)
➥0:00:00.00
system.sysUpTime.0 = Timeticks: (3601306) 10:00:13.06
system.sysContact.0 = Me <me@somewhere.org>
system.sysName.0 = hephaestus
system.sysLocation.0 = Right here, right now.
system.sysServices.0 = 72
system.sysORLastChange.0 = Timeticks: (0) 0:00:00.00
system.sysORTable.sysOREntry.sysORIndex.1 = 1
system.sysORTable.sysOREntry.sysORIndex.2 = 2
system.sysORTable.sysOREntry.sysORIndex.3 = 3
system.sysORTable.sysOREntry.sysORID.1 = OID:
➥.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB
system.sysORTable.sysOREntry.sysORID.2 = OID:
➥.iso.org.dod.internet.snmpV2.snmpModules.snmpM2M
system.sysORTable.sysOREntry.sysORID.3 = OID:
➥.iso.org.dod.internet.snmpV2.snmpModules.snmpVacmMIB
system.sysORTable.sysOREntry.sysORDescr.1 = The Mib module for
➥SNMPv2 entities.
system.sysORTable.sysOREntry.sysORDescr.2 = The Manager-to-Manager
➥MIB module.
system.sysORTable.sysOREntry.sysORDescr.3 = View-based Access
➥Control Model for SNMP.
system.sysORTable.sysOREntry.sysORUpTime.1 = Timeticks: (0) 0:00:00.00
system.sysORTable.sysOREntry.sysORUpTime.2 = Timeticks: (0) 0:00:00.00
system.sysORTable.sysOREntry.sysORUpTime.3 = Timeticks: (0) 0:00:00.00
End of MIB
```

Intrusion detection

by Lance E. Spitzner

Your network is being scanned for vulnerabilities. This may happen once a month or twice a day; regardless, there are people out there probing your network and systems for weaknesses. I can say this with confidence because I have yet to work on a network that hasn't been probed. My personal network of six systems at home is on a dedicated ISDN line. This network has no valuable data, nor represents any organization, yet it gets probed two to four times a week. If you have a system or network connected to the Internet, you become a target.

In this article, we'll discuss how you can protect yourself by detecting these intrusion attempts. We'll then cover what you can do when you discover these attempts.

Setting up intrusion detection

The methods we'll be discussing are simple in use and implementation. For larger or more security-conscious organizations, you may want to consider third party Intrusion Detection Systems, such as Network Flight Recorder (**www.nfr.net**). These more advanced IDS systems use traffic analysis and advance algorithms to determine if a probe has been conducted. Our approach will be somewhat simpler.

Listing A: *Intrusion detection alert*

```
Subject: ### Intrusion Detection Alert ###

You have received this alert because the network
is potentially being scanned. The information below
is the packet that was logged and dropped.

Date:           Sat Jan 24
Time:           18:47:46
Source:         ICARUS.CC.UIC.EDU
Destination:    lisa
Service:        imap

--- Finger Results ---
 [ICARUS.CC.UIC.EDU]
 Login       Name                 TTY      Idle
 Spitzner    Lance Everett Spitzn pts/72
 When        Where
 Sun 18:42   lspitz-4.soho.entera
```

Port scans

There are a variety of different probes hackers will attempt. The first type we'll prepare for is one of the most common—port scans. *Port scans* are where an individual attempts to connect to a variety of different ports. The scans can be used on a specific target, or used to scan entire IP ranges, often chosen at random. This is one of the most popular information gathering methods used by hackers today, as it identifies what ports and services are open.

To detect these scans, we'll build a system that alerts us via email whenever someone connects to a predetermined port. First, we identify three to five of the most commonly scanned ports. Then we select two to three systems to listen on these ports. When an intruder scans our network, he'll most likely hit our systems listening on these ports. When these ports are scanned, the systems log the attempt, execute various predetermined actions, and then email an alert to a point of contact.

The end result is you receive an email for each port scanned. If you have three systems, each listening on four ports, then you may get up to 12 emails from a single network port scan. However, this isn't normally the case. If hackers are scanning an entire network, they're normally looking for a single vulnerability, such as imap (port 143). In this case, we'd have received only three emails, one from each system. When they scan a single target, often they scan a range of ports, such as 1-1024. In that case, we'd have received only four emails—one for each port on the system. Based on the emails you get, you can quickly determine what the intruder is interested in, as shown in **Listing A**.

Implementation

To implement this methodology, we first identify two to three systems to use for monitoring. I often select DNS servers, as these are primary targets. Many scanning tools start by scanning Name Servers to build databases of IP addresses. Then, select three to five of the most commonly scanned ports. Ensure that you're not using these ports; otherwise, every time someone connects to it, you'll be alerted. To identify commonly scanned ports, CERT alerts are a

great place to start. You can find these alerts at www.cert.org. The ports we'll be using are:

- imap (port 143)
- SMB (port 139)
- login (port 513)
- http (port 80)

I like these ports since hackers commonly look for them, but most of your systems won't be using them. Make sure these ports aren't already blocked by a screening router or a firewall. We'll then set several systems to listen on these ports, alerting us when there's a connection.

TCP Wrappers

Our implementation uses TCP Wrappers. Created by Wietse Venema, TCP Wrappers allow us to control, log, and most importantly, react to any wrapped service. When someone connects to one of the services we defined, TCP Wrappers will log the connection (via syslog) and then spawn our alerting mechanism.

For those of you who don't already have TCP Wrappers installed, I highly recommend it. It's extremely easy to compile, configure, and implement. You can find it at many tool repositories, such as

ftp://coast.cs/purdue.edu/pub/tools/unix

Before you compile it, enable language extensions in the Makefile (this greatly enhances its configurability). We'll be using this capability for intrusion detection purposes. For more information on installing TCP Wrappers, I recommend you review the article, "Armoring Solaris," in our December 1998 issue.

Once we've compiled and installed TCP Wrappers, we'll want to wrap the four ports we defined. The ports are first defined in /etc/services and then added to the /etc/inetd.conf file. Here's an example of wrapping imap in the file /etc/inetd.conf.

```
imap stream tcp nowait root
➥/usr/local/bin/tcpd imap.trap
```

When someone connects to port 143, tcpd accepts the connection from `inetd`. It then looks at the /etc/hosts.allow file for access control. This is where we define what connections are allowed to launch the alert. Finally, it will finish by launching imap.trap. You'll need to rename imap.trap for each respective service, such as *http.trap* for http or *smb.trap* for smb. Following is an example of the entry in /etc/hosts.allow, this is the entry that alerts us of a possible probe:

```
imap.trap: ALL: spawn (/var/adm/ids.sh %d
➥%h %H)
```

This tells `tcpd` to accept all connections to port 143 regardless of IP, and then spawn our intrusion detection script—the script that alerts us. We want spawn instead of twist, because twist uses the remote client for all `stdout` and `stderr`. The three expansions following the ids.sh file (defined by TCP Wrappers) become in-line variables.

The script /var/adm/ids.sh is where all the action happens. You can modify it to suit your own personal taste. I've included an example that parses the data, does a `safe_finger` on the client, emails an alert to a point of contact, and optionally launches snoop to track any additional action, as shown in Listing B on page 14.

Now, whenever someone connects to one of our predetermined ports, we receive a formatted email with all the critical data. For example, a user scans our network for port 143 looking for `imap` vulnerabilities. Three of our systems are listening on that port. The connection is made, and `tcpd` is launched. It looks at /etc/hosts.allow, and finds an entry for imap.trap. It spawns our intrusion detection script /var/adm/ids.sh, which parses the data, fingers the client, and then emails an alert. We also have the option of launching tools, in this example snoop. The last thing that happens is that `tcpd` attempts to launch /usr/sbin/imap.trap, which it doesn't find. `Tcpd` then exits, logging an error to syslog. To avoid this, you may want to create a shell script /usr/sbin/imap.trap, which does nothing but exit out.

One thing to keep in mind is Denial of Service attacks. The more you have your script do, the more system overhead you incur. An attacker could disable your system by making multiple connections to the predetermined ports, thereby creating multiple processes of your scripts. I recommend that if you implement a variety of actions in your scripts, that you limit the number of processes per source IP address.

Router logs

An alternative to using TCP Wrappers is router logs. Many of us don't have the luxury of using three systems for intrusion detection. However, you can use the methodology described using your Internet router.

Once again, you select two or three systems and three to five ports to be monitored. Build an ACL (Access Control List) on your router that denies the specified ports and systems. Have this ACL log all connection attempts to a syslog server. Now you can monitor any denied traffic and quickly determine if your network has been probed. I've had great success implementing this with Swatch, which automates both the filtering and alerting processes.

These solutions aren't foolproof. Many of today's port scanners don't complete the TCP SYN/ACK sequence during a connection. In fact, many scans use invalid packets (such as FIN or Xmas scans). The methods I've discussed will *not* detect some of these scans. For more robust intrusion detection, you'll need more advanced tools.

Listing B: *Our ids.sh script to control tcpd*

```
#!/bin/ksh
#
# Script launched by tcpd for intrusion detection purposes
#

USER=lance@spitzner.net
SRV=`echo $1 | cut -f1 -d.`
DATE=`date "+%a %b %e"`
TIME=`date "+%T"`
FINGER=`/usr/local/bin/safe_finger @$2`
MAIL=/usr/bin/mail

$MAIL $USER <<EOF
Subject: ### Intrusion Detection Alert ###

You have received this alert because the network
is potentially being scanned.  The information below
is the packet that was logged and dropped.

Date:           $DATE
Time:           $TIME
Source:         $2
Destination:    $3
Service:        $SRV

--- Finger Results ---
$FINGER

EOF

##### If the service is imap, lets go ahead and snoop the
##### session.
if [ $SRV=imap ]; then
   snoop -v -c 5000 -o /var/adm/$2_snoop.$$ $2  &
fi
```

Other ways

There are other ways you can implement intrusion detection on your system. Once again, you have to first identify the intrusion methodology, and then implement a tracking and alerting procedure. An example would be brute-force attempts to log on. Five consecutive failed attempts to log on are logged in the file /var/adm/loginlog. This would happen when a hacker is probing your system for weak log on and password combinations.

I set all my systems to run a daily cron job and see if there are any entries in the file. If there are, someone has either forgotten his password and is guessing what it is, or a potential hacker is attempting a brute-force entry. The cron job emails me the entries, makes a copy to an archive, and then clears the log. Another example is the common /cgi-bin/test-cgi attack used on Web servers. Instead of disabling this cgi script, I alter it to log and email me whenever someone attempts this exploit. This usually involves nothing more than modifying the shell script test-cgi (be sure to test this before you implement it on your system).

As we've covered, there are a variety of simple ways to implement some basic intrusion detection. Though not foolproof, these methodologies will help you identify potential probes and protect your network. Now, once you've implemented intrusion detection, what do you do when you discover your systems are being probed?

Reacting to an intrusion

The first step is confirming that your systems are truly being probed. Just because you receive one email alert from our TCP Wrapper setup, does *not* mean you're being scanned. A confused user may be connecting to the wrong system, or someone is simply mis-typing a key. Nothing is more embarrassing then accusing someone of something they didn't do. However, if you have three consecutive systems scanned on the same port at the same time, this indicates that you may have been probed. Now what?

The last thing you want to do is send out a counterattack on the system and take them off the air. Retaliation of any kind isn't the an-

swer. There are a variety of reasons for this, which I'll get to in a moment. Instead, I recommend you get some information on the source, and then contact the person or organization responsible for the network.

When your network gets scanned, you may feel frustrated and want to take out that frustration on the system that probed you. Since someone is preparing to hack you, shouldn't you act? However, you want to be very careful how you react:

- Your systems may have indeed been scanned, but by accident. Large organizations often scan their internal networks and remote offices. Someone may have scanned the wrong network (I personally know of this happening at one organization).
- Often, the people responsible for the systems that scanned you have no idea of what happened. Large systems with hundreds of users may have a malicious user who is illegally using his or her account to probe other networks. Or, the system may have been hacked and is being used as a launching point. Either way, the admin of the system will want to know so they can fix the problem.
- The source IP address showing in your logs may not be a valid system; rather it may be a decoy source. Many scanning tools allow the user to change the source IP address to whatever the user wants. Your logs may show your systems were scanned from five different sources, when you were actually scanned five times by the same machine. The user is attempting to deceive the true source of the probe by using fake source IPs. It's now extremely difficult to determine which one of the scans was the actual probe. Also, the user could have faked his source IP address to lay blame on someone else.

About our contributors

Werner Klauser is an independent UNIX consultant working near Zurich, Switzerland. While not paragliding, or roarin' around on his Harley chopper, he can be reached by email at **klauser@klauser.ch** or on his Web page at **www.klauser.ch**.

Don Kuenz programs for Computing Resources Company in Casper, WY. You can find out more at **gtcs.com/crc**.

Alan Orndorff has been working with computers since 1990. He's using Solaris as a platform for Lotus Notes and at home in his spare time. He currently lives in San Francisco and can be reached at **dwarf333@hotmail.com**.

Lance E. Spitzner enjoys learning by blowing up his UNIX systems at home. Before this, he was an Officer in the Rapid Deployment Force, where he blew up things of a different nature. You can reach him at **lspitzner@enteract.com** or **www.enteract.com/~lspitz**.

Paul A. Watters is a research officer in the Department of Computing at Macquarie University, Australia. He can be reached at **pwatters@mpce.mq.edu.au**.

Inside Solaris™
Tips & Techniques for users of Sun Solaris

Inside Solaris (ISSN 1081-3314) is published monthly by ZD Journals 500 Canal View Boulevard, Rochester, NY 14624.

Customer Relations

US toll free ..... (800) 223-8720
Outside of the US ..... (716) 240-7301
Customer Relations fax ..... (716) 214-2386

For subscriptions, fulfillment questions, and requests for group subscriptions, address your letters to

ZD Journals Customer Relations
500 Canal View Boulevard
Rochester, NY 14623

Or contact Customer Relations via Internet email at zdjcr@zd.com.

Editorial

Editor ..... Garrett Suhm
Assistant Editor ..... Jill Suhm
Copy Editors ..... Rachel Krayer, Christy Flanders, Taryn Chase
Contributing Editors ..... Werner Klauser, Don Kuenz, Ian Orndorff, Lance E. Spitzner, Paul A. Watters
Print Designer ..... Melissa Ribaudo
Cover and Content Design ..... Lance Teitsworth

VP of Content Development ..... Ed Passarella
VP & Publisher, Content Development ..... Linda Edwards
Executive Editor ..... Kent Michels
Circulation Manager ..... Jeff Marcus
Manager of Design & Production ..... Stacy Dobles
Manager of Design Services ..... Charles V. Buechel
Executive VP of Operations ..... Jerry Weissberg
Director of Customer Support ..... Douglas Neff
Director of Operations & Fulfillment ..... Kress Riley

You may address tips, special requests, and other correspondence to

The Editor, *Inside Solaris*
500 Canal View Boulevard
Rochester, NY 14623

Editorial Department fax ..... (716) 214-2387

Or contact us via Internet email at sun@zdjournals.com.

Sorry, but due to the volume of mail we receive, we can't always promise a reply, although we do read every letter.

Postmaster

Periodicals postage paid in Louisville, KY.

Postmaster: Send address changes to

Inside Solaris
P.O. Box 92880
Rochester, NY 14692

Price

Domestic ..... $99/yr ($9.00 each)
Outside US ..... $119/yr ($11.00 each)

Our Canadian GST# is: R140496720. CPM# is: 1446703.

Back Issues

To order back issues, call Customer Relations at (800) 223-8720. Back issues cost $9.00 each, $11.00 outside the US. You can pay with MasterCard, VISA, Discover, or American Express.

2096

PERIODICALS MAIL

Sun Technical Support
(800) 786-7638

C#7661905 00002096 04/00
[redacted]
CLINTON TOWNSHIP, MI 48035-4218

Please include account number from label with any correspondence.

Even with the best of intentions, you can do more harm than good. For example, let's say you discover that the system that scanned you has been hacked and is being used as a launching point. You identify a backdoor the hacker left, gain access, grab all of his tools and logs, and then proceed to notify the system owner and various emergency response organizations. Even though you think you've done the right thing, you've caused more harm than good:

- Most likely the hacker replaced various monitoring tools and logs on the compromised system. He may discover you were there, and then wipe the system clean to cover his tracks (thus destroying the machine).
- The system admin may have known about the hacker and was working with law enforcement. You've just messed up their investigation.
- You can be held liable for the hacking incident. The system owners don't know you and may accuse you of being the original hacker, attempting to protect yourself by blaming someone else.

Basically, there's a lot that can go wrong and not much that can go right when you act on your own. The best thing you can do is get as much information as you can. Identify any logs that show probes from the source address. Then, identify the individuals and/or organization responsible for the incident. Next, email them with details of what happened and when, including log entries for verification. You may also want to courtesy copy the organization's upstream provider to keep them informed.

If the intrusions are serious enough, contact professional response organizations, such as CERT **www.cert.org** or CIAC at **www.ciac.org**. If the intrusion attempts continue with no response from the system owners, call the organization. The phone can be a very powerful tool.

Conclusion

Sooner or later, your systems and networks may be probed for various vulnerabilities. By implementing some of the basic measures we've discussed, you'll be better prepared to log and identify these attempts. Once identified, you can track these probes and gain a better understanding of the threats to your network and react to these threats.

When identified, it's best to gain as much information as possible, and then notify the individuals and organization responsible for the system. Taking action on your own could become messy, causing more harm than good. By working with others, you'll come to a better a solution.

Coming up...

- Server security
- More Y2K coverage