ZD JOURNALS

Volume 6, Number 5 • May 2000
www.zdjournals.com/sun
US $11.00

Inside Solaris™

Tips & Techniques for users of Sun Solaris

Roll out your own high availability

by Daniel R. Speciale

The IT managers of today expect more out of their production servers. They want highly available server configurations in order to support their mission-critical applications or to run their day-to-day operations with few outages.

The term *highly available* or HA usually means a system will be available to run your applications 99 to 99.999 percent of the time. Sometimes it's difficult to design both a cost-effective and easily maintainable HA solution. The HA products that lead in the Sun Solaris arena are Veritas FirstWatch/Veritas Cluster and Sun Cluster. These products have good functionality and keep your systems up, but their cost and administration time are sometimes overwhelming for the customer.

In this article, we'll share an approach we've used that provides reasonable or *pretty good* HA with the configuration shown in **Figure A**. This approach is working for our current customers. The primary design goal is to provide sufficient redundancy in order to maintain the target system's and resident application's availability to users.

What's pretty good HA?

Pretty good HA is the hardware and software framework of a cluster configuration without the HA software and automatic fail-over features. This provides the customer with two systems running separate applications and the ability to have either system take over the other's application in the event of a failure.

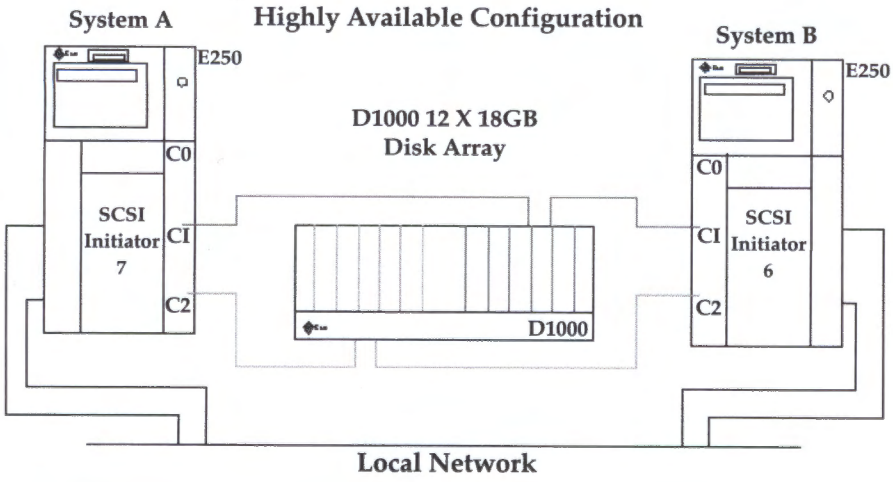

Figure A: *This is our sample configuration with two Sun E250s.*

In this issue

You have the flexibility to either build simple scripts to have a system take over when the other system goes down, or manually switch an application over to a healthy system.

The hardware HA configuration we'll discuss in this article uses two Sun E250s with a shared disk sub-system (Sun D1000). Each E250 has two differential SCSI controllers, two Ethernet interfaces, one internal hard drive, and N+1 power supplies. The software installed is Solaris 2.6 and Veritas Volume Manager. This provides sufficient hardware redundancy in the event of a component failure.

Disk sub-system selection and configuration

The key to any HA solution is the availability of data. In the configuration shown in **Figure A**, we've used two Sun E250s—System A and System B—connected via 4 SCSI cables to one Sun D1000 disk sub-system.

Keep in mind that you can use many different types of disk sub-systems for an HA solution. We picked the D1000 because it has two SCSI busses and supports two connections per bus (two from each system). The D1000 also has redundant power supplies, and it can run in a split configuration (for example, six drives on one SCSI bus and six drives on the other).

The D1000 is fully loaded with 12X18-GB disk drives that provide 45 GB of usable disk space for each system. Since System A and System B share the two SCSI busses (on the D1000), both systems can see all drives on both busses. Consequently, both systems see each other's disk controllers, C1 and C2. Since each device on a SCSI bus must have a unique SCSI address, we had to change System B's disk controller's C1 and C2 addresses from 7 (default) to 6. We made this change in the OBP (Open Boot PROM) by writing a NVRAM script to set the SCSI initiator address to 6. For additional information on the `nvedit` commands, see **Table A**.

The following are the required boot `PROM` commands for changing the SCSI initiator address to 6 on all SCSI controllers on an E250, except controller 0. After powering on the system, on the system console, type a `Stop-a` command on a Sun keyboard or a `send break` command at an ASCII terminal. You should get an `ok>` prompt. At the prompt, type the following on System B only:

```
ok> setenv scsi-initiator-id 6
```

The internal SCSI controller (C0) address will retain its default ID of 7 until the nvramrc script is entered, as shown below. We now need to determine the path for the internal SCSI bus and enter the nvramrc script to set its SCSI bus ID also to 7. Enter the following command to see your SCSI bus addresses:

```
ok> probe-scsi-all<cr>
```

Look for the internal SCSI disk drives. The path may be /pci@6,4000/scsi@3 for the E250 servers (the path may be different for other architectures). The key is to find the internal boot drive and copy the path listed directly above it.

Careful! In the sequence below, the single space after the double quotes and before `scsi-initiator-id` isn't a typo, so be sure it is there. Use the following to create the nvramrc script:

```
ok> nvedit <cr>

0: probe-all <cr>
1: cd /pci@6,4000/scsi@3 <cr>
2: 7 encode-int " scsi-initiator-id"
➥property <cr>
3: device-end <cr>
4: install-console <cr>
```

Table A: *Commands used with nvedit*

Keystroke	Description
`control-b`	Moves backward one character.
`control-c`	Exits the editor and returns to the Openboot command interpreter. The temporary buffer is preserved, but it isn't written back to NVRAMRC. (Use nvstore afterwards to write back the temporary buffer.)
`control-f`	Moves forward one character.
`control-k`	Joins the next line to the current line.
`control-l`	Lists all lines.
`control-n`	Moves to the next line of the NVRAMRC editing buffer.
`control-o`	Inserts a new line at the cursor position and stays on the current line.
`control-p`	Moves to the previous line of the NVRAMRC editing buffer.
`Delete`	Deletes the previous character.
`Return`	Inserts a new line at the cursor position and advances to the next line.

```
5: banner <cr>
6: <crtl-c>  (note- use control C to exit
➥the script as shown)
```

Now, commit the script and other changes to Non-Volitale Memory (NVRAM):

```
ok>nvstore <cr>
ok>setenv use-nvramrc? true <cr>
ok>reset-all <cr>
```

When the server completes the reset, use the `printenv` command to display the SCSI initiator ID. This verifies that the change took place.

Veritas implementation

In this HA configuration, we used Veritas Volume Manager VxVM to implement RAID 0, 1, 0+1, 3 and 5 on Solaris systems disks. The first thing we did was mirror (RAID 1) the OS disk to another disk on a second controller, because we didn't want the system going down due to one bad disk or controller.

Then, we set up volumes and filesystems with Volume Manager and put them into disk groups. All our volumes and filesystems were striped and mirrored (RAID 0+1). Volume Manager enables us to place volumes and file systems into groups. The ability to deport and import these disk groups between Systems A and B is the true power of Veritas.

HA scenarios

System A is an NFS server with a disk group called *nfsdg*, which contains the shared NFS filesystems. If System A went down, System B could simply import the disk group *nfsdg* and mount all the volumes within the disk group. You would then configure the stand-by network interface with System A's IP address, and System B now becomes your NFS server. Since the disk groups can be passed between the two systems with ease, you can use this feature to move applications from one system to another.

Let's assume that System B has two small databases. One of the databases resides in a disk group called *db10dg*. Suppose you decide to move one of the databases to System A. With the database binaries already in place on System A, you shut down the database on System B and deport the disk group *db10dg*. The disk group is then imported on System A. Next, all volumes and filesystems are mounted and the stand-by network interface is configured with the appropriate IP address on System A. Finally, you can start the database on System A.

Listing A: *Script to run on System B in the event that System A goes down*

```
#!/bin/csh
# This script runs on system B to takeover System A's
# NFS service.
# First Import System A's nfs Disk Group

vxdg import nfsdg

vxrecover -g nfsdg -sb

# Now mount the File System to be shared

mount -F vxfs /dev/vx/dsk/nfsdg /disk_share

# Next, NFS share the File system

share /disk_share

# Configure HME1 with System A's IP Address

ifconfig HME1 inet 192.9.200.1 netmask \
255.255.255.0 broadcast 192.9.200.255 up
```

Example script

You can easily write a script to do the tasks in the previous examples. Listing A displays a script that you can run on System B in the event that System A goes down.

Caveats

This HA configuration is only as good as you make it. With scripts and the correct configuration, you can build a solid HA framework.

Also, be aware that Sun doesn't officially support SCSI initiator mode outside of their Cluster product. Nevertheless, in our experience, it has worked without any issues.

Conclusion

While this HA configuration can't guarantee 99 percent up time, it's good utilization of the customer's hardware and software. If the customer decides to purchase one of the HA products we mentioned in this article, they can achieve higher levels of availability and the cluster framework is already in place and ready to go.

Synchronizing computer clocks

by Don Kuenz

Local Area Networks (LANs) function better when all computers connected to it have synchronized clocks. For instance, some security mechanisms require that the time difference between servers and clients be less than five minutes. You also need to synchronize clocks on your file server and client in order to prevent some incremental compilers from redundantly compiling untouched source files during each build.

Beware of clock drifts

First, a few words of caution before we start. This article focuses on synchronizing the clocks of computers connected to a LAN with intermittent Internet connectivity (that is, dialup). We want each host to use the same, identical time. To do so, we force each host to set its time to the value broadcast by our NTP server. When our server's clock drifts, the other hosts' clocks drift by the same amount because of this synchronization. On a LAN isolated from the Internet, you must correct for a drift by manually invoking the `date` command or by using an external reference clock. As long as you correct your NTP server's time about once a week, this configuration works for a LAN with an intermittent connection to the Internet.

In this article, we'll show you how to set up an NTP server that uses its system clock as a time reference. This configuration requires that you use the `date` command to manually correct your server's clock after it drifts. As long as you correct your NTP server's time about once a week, this configuration works for a LAN with an intermittent connection to the Internet.

Public NTP servers or reference clocks

If you enjoy a permanent connection to the Internet, you can either use public NTP servers or you can invest in a reference clock. Dozens of companies sell acceptable reference clocks. Most of the budget clocks receive time signal broadcasts from national standard agencies. For instance, you can connect one of several Global Positioning System (GPS) receivers to your NTP server's serial port and use it as a reference clock.

Take a look at **www.ntp.org** for more information about public servers and reference clocks, as well as general information about NTP. This Web site focuses on a free NTP package that runs on multiple platforms, including Solaris. For the purposes of this article, we'll stick with the NTP package that comes bundled with Solaris. However, you can still use **www.ntp.org** as a reference, because both packages share almost identical core functionality.

NTP uses the notion of an integer *stratum* to rank the quality of time offered by a reference clock. The lower the integer is, the higher the quality will be. A caesium clock, similar to those employed by the National Institute of Standards and Technology (NIST), offers the highest quality, so we named it a *stratum 0* time reference. Unfortunately, the cost of a caesium clock forces most people to use a stratum 1 reference, such as a GPS receiver.

For this article, we'll use the NTP server's system clock as a stratum 12 reference. NTP coordinates time throughout the entire Internet, which means that your NTP server can adversely affect other servers and clients. Make sure you thoroughly research NTP before you increase the stratum of your server.

Installing NTP

You need to obtain and install the necessary NTP software packages for each operating system in your LAN. Sun bundles NTP client and server functionality with Solaris. Microsoft bundles NTP client functionality with Windows 9x and Windows NT. Microsoft optionally offers NTP server functionality for Windows NT Server. The free NTP package available at **www.ntp.org** includes client and server functionality for BSD and most other UNIX operating systems.

Solaris distributes its NTP functionality in two packages named `SUNWntpr` and `SUNWntpu`. You can use `pkginfo SUNWntpr SUNWntpu` to verify the installation of these packages. If you receive an error message, you need to install `SUNWntpr` and `SUNWntpu` from your Solaris media using the `pkgadd` command.

Installing a Network Interface Card (NIC) into Microsoft hosts enables NTP client functionality. Microsoft includes server functionali-

ty only with its Windows NT Server Resource Kit. You must purchase this kit from Microsoft. Why do we mention using the NTP server functionality of NT, since we plan to use Solaris as the server for our LAN? It turns out that the usenet community warns of a bug in Microsoft's Windows 9x NTP client functionality that allows it to only synchronize with a Windows NT NTP server. So, we use Solaris to synchronize our NT host, which in turn synchronizes our Windows 9x clients. Alternatively, the usenet community says that you can synchronize your Windows 9x clients directly to a Solaris NTP server by using a free package available at

ftp://zdftp.zdnet.com/pub/private/sWlIB/utilities/desktop_accessories/atomtm.zip

You can obtain a free, precompiled NTP package for OpenBSD at

ftp://anonopenbsd.cs.colorado.edu/pub/OpenBSD

Now move to your current OpenBSD version directory. Look for the packages directory, and then the appropriate platform. Locate a package named xntp*. After you obtain the package, install it with the following command:

```
pkg_add xntp*
```

Creating NTP servers

First, let's review how Solaris interfaces with your server's system clock. You set your local time zone using a configuration file named /etc/default/init. You can also access that same file by using a handy link named /etc/TIMEZONE. Look for a single line that contains something like the following:

```
TZ=US/Mountain
```

This line tells Solaris to set its time zone to US Mountain Time. You can use the following command to display a list of all time zones currently defined on your server:

```
ls /usr/share/lib/zoneinfo
```

After you set your time zone, you need to check, and possibly set, your server's time. Fortunately, the `/usr/bin/date` command performs both functions. To check your server's clock, simply invoke `/usr/bin/date` without arguments, and it will display the system date and time. You can use either a time or a date-time argument when you wish to set your server's clock. As we said earlier, you need to check and set your NTP server's clock at least once a week unless you obtain an external reference clock. The following are examples of the various forms of the `/usr/bin/date` command:

```
# date
Wed Jan 26 18:30:37 MST 2000

# date 1832
Wed Jan 26 18:32:00 MST 2000

# date 012618332000
Wed Jan 26 18:33:00 MST 2000
```

The first date command displays the current date and time. The second command sets the time to 18:32:00. The third command sets the date to Jan 26, 2000, and the time to 18:33:00.

After you finish verifying your server's time, create a file named */etc/inet/ntp.conf* that contains the following two lines:

```
# LCL, local clock
server 127.127.1.1
# increase stratum
fudge  127.127.1.1 stratum 12
```

These two lines tell NTP to use our server's system clock as a reference, and give it a stratum of 12. The high stratum ensures that foreign NTP clients on the Internet ignore our time broadcasts when we connect to the Internet.

On Solaris, a daemon named /usr/lib/inet/xntpd implements NTP functionality for servers. The /etc/rc2.d/S74xntpd script starts xntpd whenever the system boots. You can also manually start xntpd with the following command:

```
/etc/rc2.d/S74xntpd start
```

Instead of a daemon, Windows NT uses a *service* to implement NTP server functionality. First, copy the TIMESERV.EXE and TIMESERV.DLL files from the resource kit to the %SYSTEMROOT%\SYSTEM32 directory. Next, create a file named *%SYSTEMROOT%*\

SYSTEM32\TIMESERV.INI that contains something similar to the following:

```
[TimeServ]
Type=NTP
Period=0
NTPServer=192.168.10.1
timesource=no
Log=no
```

Our NTP server uses an IP address of 192.168.10.1. You need to change the `NTPServer` parameter to the IP address of your own NTP server. After you finish editing TIMESERV.INI, open a command prompt window and invoke the following commands:

```
timeserv -update
timeserv -automatic
```

Reboot your NT host, and it will start functioning as an NTP server for Windows 9x clients.

Creating NTP clients

On a Solaris, OpenBSD, and most UNIX clients, you can invoke the following command to set a client's clock to the time broadcast by your NTP server:

```
/usr/sbin/ntpdate 192.168.10.1
```

Our NTP server uses an IP address of 192.168.10.1. You need to change that argument to the IP address of your own NTP server. You can automatically set the date during boots by invoking `ntpdate` from within a startup script. If your clients run all of the time, you can use cron to periodically invoke `ntpdate`.

On Windows clients, you can invoke the following command to set the client's clock to the time broadcast by an NT NTP server:

```
NET TIME \\NTSERVERNAME /S /Y
```

`/S` tells Windows to set the time, while `/Y` automatically answers "YES" to the prompt "Are you sure?" This allows you to invoke the command without bringing things to a halt until someone answers.

Conclusion

NTP allows you to synchronize the clocks of all computers connected to a LAN. You can use the NTP functionality that Sun bundles with Solaris to create a server that broadcasts time information to NTP clients on a LAN. Clients can invoke either `ntpdate` or `net time` to obtain and set their system clocks to the time broadcast by NTP servers.

My file system is full! Now what?

Nothing is worse than running out of space on one of your partitions. If you're using products like Veritas Volume Manager, you can dynamically grow UFS file systems. But if you're not, you have to do it the old-fashioned way. In this article, we'll discuss the strategy we used to fix our full /usr partition. Our disk space problem also gave us an excuse to buy more hardware, which is always a good thing. Listing A shows our one SCSI drive (using df –k) with a /usr partition that's quite full.

Add a new disk to your system

The first step for making room on your file system is to add a new disk to your system. This may be a SCSI, Fibre Channel, or IDE based on your system's device interconnects. In our case, we added a nice Quantum Viking II SCA drive that we purchased for under $200 off the Internet. Our target box is a Sparc 20, so heat and power considerations were very important.

Our new disk has a SCSI ID of 1 on the system to be upgraded. You can find your disk's ID by checking the /dev/dsk directory. The device that's listed yet not mounted (shown via df –k) will be the new drive.

Now that we've installed a new drive, we need to prepare it for use. The first step is to format the drive and then create a new file system (via `newfs(1M)`).

Format the drive

The `format(1M)` command gives you a list of valid disks on your system. To access this list, you first need to log on as root, and then choose your new disk from the list. Format now displays a list of possible commands. Typing `help` shows the list of available commands. You may choose to run the `analyze` command before you actually perform the format to ensure the integrity of your disk. This command will present a list of tests that you can use to check your disk. Type `exit` to get back to the format menu.

At this point, you'll now format the empty disk by using the `format` command. The `format` command tells you how long it estimates this operation will take and asks if you want to continue. Make sure the disk doesn't have anything on it that you want to keep, because it will be completely erased.

The next step is to partition the disk according to your needs. Type `partition` at the format prompt to start the partition manager. A new menu of options appears. You can use the `print` command to see the current partition layout. Our layout is shown in **Listing B**. We wanted to let our /usr partition take the whole disk, so we allocated all available space to one slice. If you don't like your existing layout, then you can modify specific slices or use the `modify` command to build a new layout with the current layout as your starting point. Once you're happy with your layout, exit the partition manager and the format program.

Create a file system on your formatted disk

Now you have to create a file system on your newly formatted disk. You can do so with the `newfs(1M)` command. Simply run the command for each disk slice that you want to make available. Because we only have one, we just need to run the command once:

```
newfs /dev/dsk/c0t1d0s7
```

There are also options available to customize how `newfs` creates your file system. The most important is the `-i` parameter, which controls the number of inodes created. The value increases the number of bytes allocated per inode, and defaults to 2048.

Listing A: *Our /usr file system as shown with df –k. It is definitely in need of some relief.*

```
/proc                    0       0       0     0%    /proc
/dev/dsk/c0t3d0s0   336879   63941  239251    22%    /
/dev/dsk/c0t3d0s6   649711  584596    6642    99%    /usr
fd                       0       0       0     0%    /dev/fd
/dev/dsk/c0t3d0s7  1488463       9 1428916     1%    /export/home
/dev/dsk/c0t3d0s3  5704022 1530697 4116285    28%    /u01
swap                737228     280  736948     1%    /tmp
```

Listing B: *Our new disk layout. Notice that we have allocated almost all space to slice 7.*

```
Part      Tag    Flag     Cylinders         Size            Blocks
  0 unassigned    wm       0                0         (0/0/0)           0
  1       swap    wu       0                0         (0/0/0)           0
  2     backup    wu       0 - 5147      4.24GB    (5148/0/0) 8895744
  3 unassigned    wm       0                0         (0/0/0)           0
  4 unassigned    wm       0                0         (0/0/0)           0
  5 unassigned    wm       0                0         (0/0/0)           0
  6        usr    wm       0 -    1      1.69MB    (2/0/0)        3456
  7 unassigned    wm       2 - 5147      4.24GB    (5146/0/0) 8892288
```

So if you need more inodes than the default, you'll need to *decrease* this option. This is useful if you plan to have a file-system full of an extreme number of small files. For example, if we wanted to double our inode count, we could have run the command

```
newfs -I 1024 /dev/dsk/c0t1d0s7
```

Mount and copy

Now you can mount this new drive and copy your existing /usr partition to it. To mount your new drive, use the following:

```
mount /dev/dsk/c0t1d0s7 /mnt
```

Now copy the contents of your /usr partition to the new drive. To do so, first change to the /mnt directory (`cd /mnt`) and then type the following:

```
ufsdump 0bf 126 - /usr | ufsrestore rf -
```

What we're doing is piping the output from ufsdump directly to ufsrestore. The parameters for ufsdump and what they do are shown in **Table A**.

After the pipe, call ufsrestore with the following parameters:

- r—Recursively restores all directories and files.
- f—Restores a specific file, in this case standard input as directed with the hyphen (-).

Table A: *The parameters for ufsdump*

Parameter	Function
0	Specifies a 0-level dump (all files)
b	Specifies a blocking factor
126	Represents the value we use for the blocking factor
f	Tellsufsdump to output to a file (In this case the hyphen is used to indicate stdout [-].)

Listing C: *Informational messages we received from ufsdump when moving /usr*

```
bash-2.03# ufsdump 0bf 126 - /usr | ufsrestore rf -
  DUMP: Writing 63 Kilobyte records
  DUMP: Date of this level 0 dump: Tue Jan 25 15:03:07 2000
  DUMP: Date of last level 0 dump: the epoch
  DUMP: Dumping /dev/rdsk/c0t3d0s6 (dieter:/usr) to standard output.
  DUMP: Mapping (Pass I) [regular files]
  DUMP: Mapping (Pass II) [directories]
  DUMP: Estimated 1226632 blocks (598.94MB).
  DUMP: Dumping (Pass III) [directories]
  DUMP: Dumping (Pass IV) [regular files]
Warning: ./lost+found: File exists
  DUMP: 43.39% done, finished in 0:13
./local/samba/var/locks/browse.dat: not found on volume
  DUMP: 89.40% done, finished in 0:02
  DUMP: 1226608 blocks (598.93MB) on 1 volume at 469 KB/sec
  DUMP: DUMP IS DONE
```

Listing D: *Our temporary slice after the ufsdump/ufsrestore step. Note that the discrepancy is our missing Samba file.*

```
bash-2.03$ df -k /usr /mnt
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t3d0s6     649711   586799  6638      99%    /usr
/dev/dsk/c0t1d0s7    4376206   584600  3745645   14%    /mnt
```

Once this process completes, you should see messages similar to those in Listing C. Note that we received an error with a Samba file that had disappeared after the catalog operation, but before the actual dump of the file. This isn't a major concern for us since it's a temporary file, but you should look for any errors such as these. To be absolutely safe, it would be best to place your system in single user mode (init 1) and do the ufsdump/restore step at the console.

Now make sure you're happy with the results of the ufsdump/ufsrestore step. We checked our file systems with the `df` command, as shown in Listing D, to make sure the results were what we expected.

So why use ufsdump/ufsrestore instead of tar, cp, or cpio? One reason is that it's been optimized for backups in the Sun environment and is extremely fast. Also, unlike tar, the dates associated with all of your directories and files are kept intact.

Update vfstab to point to your new partition

We're almost finished. The last step is to update your vfstab to point to your new partition. In our case, we just replaced the old slice reference with our new slice reference, as shown in Listing E.

Of course, you can solve full file systems with other approaches besides moving the complete file system to a larger partition. For instance, you could move portions to another partition and use symbolic link references to tie it all together. But a complete move to another partition is a clean and manageable solution in the long run. Also, remember that you should have a full backup of everything before embarking on a project such as this. The job you save could be your own!

Listing E: *The entry in /etc/vfstab shows our /usr mount point now using our newly created and formatted drive.*

```
#!Usr has been moved to the 2nd disk (Quantum Viking II)
#/dev/dsk/c0t3d0s6      /dev/rdsk/c0t3d0s6      /usr    ufs     1       no      -
/dev/dsk/c0t1d0s7       /dev/rdsk/c0t1d0s7      /usr    ufs     1       no      -
```

Provide the required security for your systems and data

by Edgar Danielyan

In the articles, "Defining an Acceptable Use Policy," in the February issue, and "Securing your networked systems with Solaris 7," in the March issue, we looked at computer security from both the social engineering and technical viewpoints. Both of these are important for the complete security framework, but are often difficult to combine in a natural, straightforward manner. In this article, we will consider the implications of various security techniques and tools and recommend how to use them together to provide the required level of security for your systems and data.

Integrating security

As you know, security comes at a cost, and not only at the cost of your time and effort, but also at the cost of security infrastructure, which includes hardware, software, and expertise. While you may have unlimited volumes of the latter, the first two are limited, and knowing how to integrate them well isn't an easy task.

Create a Risk Assessment Document

First, it usually pays to justify the costs involved to the management of your company or organization. To *justify*, in this context, means to let management understand that security isn't a luxury, but a basic need. To assist you in that seemingly impossible task, a Risk Assessment Document is invaluable. When creating one, define the risks you face in your network and on your systems, and provide the value of your assets and risks to these assets. Try to express values in dollar figures. Sentences like "it will be very costly" are of no particular help. Demonstrate that there's a potential threat and that it's high enough to warrant additional security measures.

A rock-solid method for doing this is to set up a passive network sniffer (packet capturer) on your network and then show the results to management. Another good addition would be to attach unsuccessful login attempt logs from your UNIX systems. You may also want to run some scanners, such as ISS, NetSonar, or cybercop from outside of your network to see what they say.

You can complete this information, in turn, by discussing the potential loss of both reputation and profits in case of a successful attack. Referring to security problems and attacks in the media may help you prove that all of this is for real. A good, but expensive, idea is to have an external consulting company provide an assessment of your situation (including the social engineering side of it).

Define your Security Mission Statement

Your next step in this endeavor is to define your Security Mission Statement. This document defines the fine line between the security requirements and ease-of-use requirements of your site. Things to consider at this stage are:

- What do your users expect with regard to your system's security and security procedures? Are they too low, too high, or just right?
- What implications will a breach of security have for your users, both individually and at the corporate level?
- How much downtime can your users tolerate?
- How many security problems have you had? Did you solve them completely?
- Are there insider threats? If so, clearly identify and document them.
- How much sensitive information is online (connected to the network)?
- Are there any existing security considerations being taken into account at your site or at the corporate level? Are they adequate from your viewpoint?
- Are the security considerations consistent with the nature of your business and the

economic situation of your company or organization?

Invoke security awareness

The next item on your to-do list is to raise security awareness. You may spend hours polishing your security framework, but an ignorant user will be able to ruin it in minutes! For this reason, a security awareness training program, however small and short, will be of great value to you.

Provide training via different methods—internal brochures, regular announcements, Web pages, classes, tutorials—there's no limit to your creativity in this area. Your goal is to have all employees understand how important security is for them and the company. Stage mock incidents to see how the staff is coping, and provide feedback. Let them know that whenever they have security concerns, they can call or email you for advice. And finally, review your approach and techniques quarterly to see whether they still meet the challenge.

Be aware of the most common security problems

You'll probably be surprised to find out that most security problems occur because sites don't allocate necessary resources or staff to handle and take care of network and system security, or that security personnel don't have the necessary support of their upper management in their endeavors. Many systems remain vulnerable to widely known exploits only because the administrator doesn't install vendor patches, which are available at no charge.

Further, most sites don't monitor their network traffic, and many sites either don't have or don't implement the most basic security procedures. And last, but not least, many people place too much trust in "security through obscurity," which, of course, is a bad idea.

Security in action

There are a number of commonly used attack methods that account for the vast majority of security problems. Here's a brief list of them:

- Exploitation of bugs in programs (such as buffer overflows, etc.)
- Poorly structured and/or written CGI scripts (on Web servers)
- Varied email-related problems, such as spamming, bombing, etc.
- Misconfigured FTP and Web servers
- Denial of Service (DoS) attacks using various methods, including the most primitive methods (Strangely enough, the most primitive ones aren't always the least effective ones.)

The most frequently replaced UNIX binaries after a successful attack are login, ls, ftpd, telnetd, ifconfig, df, tftpd, libc.a, netstat, ps, and cc. A number of other files also may be modified or replaced, such as /.rhosts, /etc/hosts.equiv, /bin/.rhosts, /etc/passwd, and /etc/group. Such tools as ASET (described in February 2000's article "Securing systems with ASET" by Alan Orndorf) may assist you in checking and keeping these files under control.

Have a to-do list. It's much easier to do something when you know that you have to. In keeping up with this age-long concept, have a to-do list for yourself and other security personnel at your organization:

- Develop and recommend internal security practices.
- Help define, write, and maintain official security documents.
- Continuously monitor and audit networks and systems for strange events.
- Review security logs daily (yes, daily) and deal accordingly.
- Test, install, and maintain security software (did you notice that *test* comes first?).
- Stay tuned to the new security threats and corresponding countermeasures.

Such host-based tools as COPS, crack, Tiger, Tripwire, logcheck, logger, NetSonar, and NOSAdmin may greatly assist you in most tasks. On the network side, TCPDUMP, ipfilter, portmap, SOCKS, tcpwrappers, SATAN, and nmap are useful in monitoring and controlling the network.

And finally, you have to somehow assess the effectiveness of your work. You should perform a new systems installation audit that checks whether the newly installed systems

are in compliance with the security framework and procedures. Further, execute regular automated checks on your production systems to see whether there are any strange files or modifications. Finally, perform account activity audits. A regular auditing program will keep everything in shape as long as you remain consistent in your endeavors.

Some low-cost security improvements are designed to configure your routers and switches to allow only the traffic you need and keep servers (especially sendmail, ftpd, and Web server) correctly configured. Also, pay special attention to the server-side, including (SSI) and Common Gateway Interface binaries (/cgi-bin/).

After attacks happen

Up to this point, we have looked at the measures intended to prevent attacks from happening. But this world isn't perfect, and attacks happen quite often. So what should you do in such a case? Here are few no-nonsense recommendations:

- Follow the established reporting line—that is, inform the related people in the order they should be informed, especially if your organization is large.
- Contact the incident response center most appropriate for your organization (if applicable).
- Communicate via an out-of-band channel (phone for example) to ensure that the information you're communicating doesn't end up in the hacker's hands.
- Document all your actions—your phone calls, modified files, configuration, etc.
- Make copies of modified files on a non-networked system.
- And, if appropriate, contact the police and the FBI for assistance if you will be seeking legal remedies. Remember that you'll have to produce evidence and it's up to you to collect it.

Common problems

The most common problems encountered when defining and implementing security measures are often not taken into account. For example, in many cases management may assume that once you've installed a firewall or a network filter, you don't have to worry about security anymore—wrong, and sooner or later it will be proved by a compassionate hacker.

Some network services, such as http, aren't filtered due to their high volume. This can be dangerous, as there are many vulnerabilities available to be exploited by www. Misconfigured and/or misplaced access lists may create a lot of problems, diminishing their value as security measures. Also, logging activity both on individual systems and the network is either absent or not sufficient. Remember that "The enemy is not sleeping!" so neither can you!

Trojan horse software and denial of service

by Clayton E. Crooks II

According to reports from CERT (www.cert.org/advisories/CA-2000-01.html), a SANS advisory, and the National Infrastructure Protection Center (NIPC) (www.fbi.gov/nipc/trinoo.htm), many Solaris computers have been infected with trojan horse software. Trojan horse software is deliberately and manually installed by malicious users at sites that are attacked. The site hackers use a variety of tools, such as Trinoo, TFN, TFN2000, or Stacheldraht (a German term for barbed wire). Unlike a virus, the trojan horse software doesn't spread without user interaction, and is only installed when

someone intentionally issues a command to install it.

The NIPC has noted that many of the victims have high bandwidth Internet connections, which could represent a possible threat to Internet traffic and thus make the trojan horse software a great concern for all Internet users. Further, the technical vulnerabilities used to install the denial of service tools are widespread, well known, and readily accessible on most networked systems throughout the Internet.

How the trojan horse works

The basics behind how the trojan horse software works are quite simple. Master computers using various communication channels control the infected machines. The infected machines are, in turn, used as a collective group of computing power to attack and shut down other sites by flooding networks or targeted systems at the request of the master machine. This ultimately results in denial of all service to or from the target systems.

Although the exact number of computers that can coexist as a group is unknown, it has been estimated that as many as 230 may work cohesively to attack individual machines. These attacks have succeeded in flooding out both large and small sites.

The trojan horse software is being installed continuously as the attackers continue coming back looking for new computers to compromise. Several sites have found it installed on multiple computers. It appears the attackers have constructed relatively good maps of the computers at the sites they are attacking to allow for easy repeat penetrations.

Test for the trojan horse

Currently, you can use two principal tools to test for the possible infection of your Solaris computers. The first tool, called find_ddos, was developed by the NIPC and can be installed on each host. You'll find this tool at **www.fbi.gov/nipc/trinoo.htm**. Dave Dittrich and Marcus Ranum developed the second tool, called gag, which you can remotely run to scan your systems. You'll find this tool at **staff.washington.edu/dittrich/misc/sickenscan.tar**. There's no charge for either tool.

During the first weeks of January, the Global Incident Analysis Center (GIAC) released an early notice, and several dozen organizations tested the NIPC software and provided feedback that helped improve the effectiveness of the tool. The results found that the NIPC software works well. Unfortunately, to run it, you'll be required to install it on every system you want to test. On the other hand, the Dittrich/Ranum program is a network scanning tool and should potentially be more efficient, since one tool can scan an entire network.

If your organization doesn't have tightly secured Solaris systems, you should probably use both tools. Doing so would increase your chances of finding the trojan horse software on your systems.

Because the trojan horse software is under constant development, the detection tools we've mentioned may be less and less effective as they're altered. As with most types of viruses or the trojan horse software, prevention is the best possible solution.

The most common paths used to compromise systems to insert the trojan horse software have been weaknesses in RPC (remote procedure call) implementation. You can protect yourself from this software by applying software patches (obtainable from **www.sun.com**) that are available to prevent the exploitation of these vulnerabilities. Along with these specific patches, you should also install all applicable security patches that Sun has released.

Although Solaris is the current focus of these attackers, they will soon turn to Windows NT, Linux, and other UNIX variants. If you have any of these additional types of machines, take this opportunity to close the holes there as well.

Note that you shouldn't just manually search for the trojan horse software, as you probably won't find it. The attackers have been diligent in hiding their work, making it difficult to track them down without the aide of software tools.

Solstice DiskSuite 4.x—tips and pitfalls, part 1

by J. D. Baldwin

Solstice DiskSuite 4.x (with the current version at 4.2) is a tool for managing the composition of logical devices (known as metadevices) with RAID techniques. You'll find an excellent introduction to DiskSuite and mirroring system partitions in the article "Using Solstice DiskSuite 4.2," in the July 1999 issue. In this article, we'll assume that you're familiar with the basics of DiskSuite operation covered in that article.

This month we'll introduce you to the concepts of the md.tab file and expanding a filesystem with DiskSuite. Next month we'll look at dynamically replacing a disk in a SparcStorage Array (SSA) and some tips and tools for automated error notification.

For purposes of the topics covered here, all 4.x versions are substantially the same. Make sure the version you use is compatible with your Solaris release, as mismatches can cause kernel panics and other failures.

The md.tab file—document the system configuration

The first thing to remember about the md.tab file's role is that, while it's an important configuration file, it isn't consulted by DiskSuite unless you execute a DiskSuite command that would be ambiguous without it. It's entirely possible for the md.tab file to be completely out-of-sync with the actual system configuration, and this won't affect system operation at all. However, the easiest and most effective way to add metadb replicas and metadevices to your system is to build the appropriate lines in md.tab and then execute the commands to create those devices.

Many Solaris admins running DiskSuite, including some fairly experienced ones, have simply never bothered to learn about md.tab. Some use it but don't keep it up-to-date. If you get into the habit of making your changes in this file and only then executing commands to change configurations, you'll have a time-saving tool that also documents your configuration.

If you're already running DiskSuite and don't have an md.tab file, or you don't have a current one, you can get an md.tab file reflecting your system configuration with the following command:

```
#  metastat -p >> /etc/opt/SUNWmd/md.tab
```

Note the location of the md.tab file. It's under /etc to ensure that it will always be available as long as the root partition is mountable. The command above appends to it so that you may edit the resulting file to preserve any previous contents, comments, etc. You'll also want to edit two-way mirrors to be one-way (read on for more about this).

Build the metadb information manually

Unfortunately, documenting the system configuration technique won't capture the metadb information, so you'll have to build that manually. We'll assume that you have DiskSuite installed (as detailed in the July 1999 article), but that you're just getting started with the configuration and haven't created any metadb replicas or metadevices yet.

Suppose you have four hard drives and you want three metadb replicas on slice 0 of each one. You might wish to break them out into groups of two, according to the controller. The md.tab entries might look like

```
# metadb entries
#
# c1 disk metadb replicas:
mddb01   -c 3  c1t0d0s0 c1t0d1s0
# c3 disk metadb replicas:
mddb03   -c 3  c3t0d0s0 c3t0d1s0
```

You can see from this example that md.tab comments may be delimited with the pound (#) character. Note that this won't actually create any metadb replicas, but it does make their creation (and deletion) simpler and self-documenting. To create the initial metadb replicas, once these entries are in the md.tab file, execute these commands:

```
# /usr/opt/SUNWmd/sbin/metadb -a -f mddb01
# /usr/opt/SUNWmd/sbin/metadb -a mddb03
```

(The -f, or force, option in the first instance is required only because no metadb replicas exist when the first one is created.)

Benefits

Is this extra trouble worth it? Sure, it will save you a little bit of time here and there, for example, in the instance when you have to take c1 offline and you can delete and restore the metadb replicas with the simple commands:

```
# /usr/opt/SUNWmd/sbin/metadb -d mddb01
...
# /usr/opt/SUNWmd/sbin/metadb -a mddb01
```

As noted, the timesavings is small (but nonzero). The self-documenting nature of this practice is probably more important. But the *real* benefit here is that the possibility of error is reduced significantly. If you have a group of metadb replicas distributed across, say, six disks (a common situation for SSA managers), it might be easy to mistype the name of one of them in a long list like that. But if they're grouped under the mddb*xx* convention, it's easier to type and verify before executing.

This benefit is realized more dramatically when working with metadevices themselves. Metadevices are specified in the md.tab file with the same syntax as the `metainit` command. For example, the entry for d0 and d1, which are the two submirrors of d2 (which is to be mounted on /), would look like

```
# d0: first submirror of d2, / partition:
d0    1 1   c1t0d0s1
# d1: second submirror of d2, / partition:
d1    1 1   c3t0d0s1
```

(We assume that c1t0d0s1 and c3t0d0s1 are of identical sizes.) As with the `metainit` command, the first item is the metadevice name (always in form d*x*), the numbers (1 1 in the example) are the number and width of stripes, and the final item is the list (a singleton, in this example) of partitions from which to build the metadevice.

Once these entries are present in md.tab, you can use `metainit` commands to create (or, if necessary later, delete) them more simply:

```
# /usr/opt/SUNWmd/sbin/metainit -f d0
# /usr/opt/SUNWmd/sbin/metainit d1
```

(This example assumes / is already mounted on c0t0d0s1, hence the -f option.) Setting up the mirror is a little trickier. Intuitively, one might want to write an entry like the following:

```
# d2: mirror for / partition
d2 -m d0 d1   # WRONG AND DANGEROUS!!!
```

Build a mirrored metadevice

Never specify a two-way mirror (which is what the previous example is) in md.tab. In the event that you must reconstruct a device; DiskSuite has no way of knowing which submirror is good and which is bad. So, when creating d2, it simply mixes and matches the contents of the two submirrors pretty much randomly. The result, of course, is garbage.

Note that the output of metastat -p will show mirrors in this way. If you use metastat -p to build your initial md.tab file, keep this in mind and edit it appropriately.

The proper way to build a mirrored metadevice is to put this line in md.tab:

```
# d2: mirror for / partition
d2 -m d0
```

Then, create the one-way mirror with the following:

```
# /usr/opt/SUNWmd/sbin/metainit d2
```

Now, use `metattach` to add the second submirror:

```
# /usr/opt/SUNWmd/sbin/metattach d2 d1
```

This is probably a good place to mention numbering conventions. Many admins who use DiskSuite use a convention where the lower metadevice number is the mirror, followed by the submirrors. (We used this convention in the July, 1999 article.) Many others prefer to number them in the order of creation—the submirrors first and then the mirror—as in the previous example. Choose whichever method makes more sense to you. Be consistent in your configuration, and recognize that when you look at someone else's configuration it might be the other way around.

Concatenated devices—expand filesystems dynamically

It happens all the time. You're out of space and everything would be just fine if only that 1-GB filesystem were 1.2 GB (or 1.5 GB, 1.8 GB, etc.) instead. You have plenty of room to spare on another disk, but no way to make use of it.

Symbolic links are one solution, of course, but not always a completely satisfactory

one. DiskSuite provides a way to expand a filesystem directly, and what's more, it won't even need to be taken offline.

Before actually expanding existing filesystems, we need to discuss the concept of a *concat* metadevice. Remember the 1 1 from the previous example? It specified that the metadevice is one stripe deep (that is, it has one stripe) and one stripe wide. In other words, it's a *simple* metadevice, built out of a simple partition.

By changing the second digit (for example, 1 6), it's possible to construct RAID-1, or striped metadevices, with dramatic performance improvements in high I/O environments. However, RAID-1 details are beyond the scope of this article.

The first digit of the pair designates the number of devices to be concatenated together to make a single metadevice. Each element of the concatenated device must have a stripe width specified. So, to concatenate two partitions together, the notation would be:

```
# d15: concatenated metadevice
d15      2 1   c1t2d0s2   \
           1   c1t3d0s4
```

(The backslash [\] character indicates line continuation.) This entry describes a metadevice built from the entire disk c1t2d0 with the space from c1t3d0s4 appended to it. Suppose c1t2d0s2 is a 2-GB disk, and c1t3d0s4 is a 1-GB slice. If we create a new filesystem with newfs and mount /dev/md/dsk/d15, the system would see a 3-GB filesystem. In writing to it, all data would initially be put in c1t2d0s2, and only when that disk was full would writing to c1t3d0s4 begin. These details are transparent to applications and users.

Of course, these sort of concat metadevices may be mirrored, just as simple ones and striped ones may. Suppose you have a mirrored, simple 2-GB metadevice already in use, and you need to add 1 GB to it for a total of 3 GB. For our example, the existing metadevice's entry in md.tab looks like

About our contributors

J.D. Baldwin is a UNIX and security consultant living in west Michigan. He is a graduate and former faculty member of the US Naval Academy in Annapolis, MD and has an M.S. in Computer Science from the University of Maryland. He can be reached at baldwin@netcom.com.

Clayton E. Crooks II is a self-employed computer consultant living in Knoxville, Tennessee. He's married with one child. His hobbies include game development, 3-D modeling, and any athletic activity he can fit into his busy day.

Edgar Danielyan is currently self-employed. His list of qualifications include Cisco Certified Network Associate, Diploma in Company Law from the British Institute of Legal Executives, and certified paralegal from the University of Southern Colorado. He has been working as a network administrator and manager of a top-level domain of Armenia. He's also worked for the United Nations, the ministry of defense, a national telco, a bank, and has been a partner in a law firm. He speaks four languages, likes good tea, and is a member of ACM, IEEE CS, USENIX, CIPS, ISOC, and IPG, to name a few. He can be reached at edd@danielyan.com.

Don Kuenz works at Computing Resources Company (gtcs.com/crc). They provide programming, administration, and hardware for Sun and PC platforms. You can reach Don at kuenz@gtcs.com.

Daniel R. Speciale is a senior systems engineer at AVCOM Technologies, specializing in high availability on the Solaris platform. He is married with two children and can be reached at daniel@avcom.com.

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 journals@zdu.com.

Editorial

Editor ..... Garrett Suhm

Assistant Editor ..... Jill Suhm
Managing Editor ..... Michelle Rogers
Copy Editors ..... Rachel Krayer
Glenna Lechner
Contributing Editors ..... J.D. Baldwin
Clayton E. Crooks II
Edgar Danielyan
Don Kuenz
Daniel R. Speciale
Print Designer ..... Melissa Ribaudo

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) 272-0064

Or contact us via Internet email at inside_solaris@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.

ZD Journals

General Manager ..... Kelly Baptiste
Manager of Customer Relations ..... Heather Loiacono
Manager of Operations ..... Cristal Haygood
Manager of Print Design ..... Charles V. Buechel
Manager of Product Marketing ..... Mike Mayfield
Senior Product Marketing Manager ..... Brian Cardona

Postmaster

Periodicals postage paid in Rochester, NY and additional mailing offices.

Postmaster: Send address changes to

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

Printed in the USA.

Price

Domestic ..... $129/yr ($11.00 each)
Outside US ..... $149/yr ($13.00 each)

Our Canadian GST# is: R140496720. CPM# is: 1446703. QST# is: 1018491237.

Back Issues

To order a back issue from the last six months, call Customer Relations at (800) 223-8720. Back issues cost $11.00 each, $13.00 outside the US. You can pay with MasterCard, VISA, Discover, or American Express.

Are you moving?

If you've moved recently or you're planning to move, you can guarantee uninterrupted service on your subscription by calling us at (800) 223-8720 and giving us your new address. Or you can fax us your label with the appropriate changes at (716) 214-2386. Our Customer Relations department is also available via email at journals@zdu.com.

Coming up...

- CDE tips
- Build your own IDS with logsurfer
- Remote administration


```
# d20: mirrored simple metadevice
d18      1 1   c1t3d0s2   # 2GB disk
d19      1 1   c3t3d0s2   # 2GB disk
d20   -m d18
```

and we wish to add slices c1t4d0s4 and c3t4d0s4 to the submirrors, respectively.

In order to accomplish the online expansion, first edit the md.tab file to reflect the new desired configuration:

```
# d20: mirrored concatenated metadevice
d18   1 1   c1t3d0s2  \       # 2GB disk
        1   c1t4d0s4          # 1GB slice
d19   1 1   c3t3d0s2  \       # 2GB disk
        1   c3t4d0s4          # 1GB slice
d20   -m d18
```

Keep in mind that changing the contents of md.tab alone does nothing to the system configuration, so skipping this step won't impact your actual configuration, except that it will then be undocumented. You must now reconfigure d18 and d19 to reflect this with a variant of the `metattach` command:

```
# /usr/opt/SUNWmd/sbin/metattach d18 c1t4d0s4
# /usr/opt/SUNWmd/sbin/metattach d19 c3t4d0s4
```

All of this is done with the filesystem mounted and still in use, without affecting system operation.

After the first command, the size of the mirror, metadevice d20, is still 2 GB; even though d18 is now 3 GB in size. But once the last submirror is increased in size (with the second command), the size of the mirror itself increases. Thus, after the second command is executed, d20 is 3 GB in size.

Even though d20 is 3 GB in size, a quick check of df -k will confirm that the filesystem on which it's mounted is still 2 GB. You must still grow the UFS filesystem.

To do so, enter the `growfs` command. `growfs` dynamically resizes a UFS filesystem (and only UFS filesystems) to be the size of the metadevice on which it's mounted. You must specify both the mount point and the raw device. If /dev/md/dsk/d20 is mounted on /export/home, for example, you can issue the following command:

```
# growfs -M /export/home /dev/md/rdsk/d20
```

Doing so expands /export/home into a 3-GB filesystem without affecting its contents. However, this part of the operation exacts a price. The filesystem will be write-locked for the duration of the expansion, which might be an hour or more for a multi-gigabyte expansion on a slow system. Sometimes this is acceptable, but if it isn't, `growfs` provides an option (-s *<sectors>*) to expand the filesystem in stages, such that I/O can continue (more slowly, of course) while the expansion takes place. Consult the growfs (1M) man page for more information.

Final note

DiskSuite is a lot like UNIX itself. It's flexible, powerful, command-line oriented—though a GUI front end exists—and occasionally (and subtly) dangerous. Next month, we'll explore dynamically replacing a disk in a SparcStorage Array (SSA) and look at tips and tools for automated error notification. ZDJ

QUICK TIP

Creating multiple subdirectory levels at once with mkdir -p

A great thing about the Solaris `mkdir` command is that you can easily create multiple levels of directories with one command. For instance, let's assume that you have a directory named /home/cad. You just gained a new customer named acme, and decide that you'd like to create your new files in an organized manner. Assuming that /home/cad is the only directory that exists, you can type:

```
mkdir -p /home/cad/customer/acme/project1
```

This will create the customer, acme, and project1 directories with one easy command.