ZD JOURNALS

Volume 6, Number 4 • April 2000
www.zdjournals.com/sun
US $11.00

Inside Solaris™

Tips & Techniques for users of Sun Solaris

Turn a Solaris box into a packet-filtering firewall

by Boris Loza

Today with the Internet growing so rapidly, it is almost impossible to avoid being connected online. But connecting to the Internet, or to any network that we don't trust, requires making security preparations. In this article, we're going to show you how to create a packet-filtering firewall with a Solaris box using a simple network configuration like the one shown in **Figure A**.

Nowadays, probably everybody has heard of firewalls and what they do. But, maybe not everybody knows that a Solaris box can be easily turned into an inexpensive, packet-filtering firewall. Before going into detail as to how to do this, let's go over what a firewall is.

What's a firewall?

Simply put, a *firewall* is a device used to control network traffic. It prevents outsiders from gaining access to your network and stops unauthorized connections from the inside.

In order to accomplish these tasks, a firewall needs to know at least the following information: a source IP address, a destination IP address, a source port, a destination port, and flags (TCP only). This information can be retrieved from a network packet header. Depending on how this information compares to the firewall access control list, the packet is either allowed to pass or it's dropped. A firewall that looks at this minimal amount of information is a *static packet filter*.

In addition to the information used by a static, packet-filtering firewall, a dynamic packet filter maintains a connection table in order to monitor the state of a communication session. This type of a firewall is able to remember previous communication packet exchanges and decide (based on this additional information) whether the session is allowed or denied.

Another firewall type is a proxy or an application firewall. With a proxy firewall source and destination, systems will never actually connect to each other. Instead, they're talking to the proxy for all connection

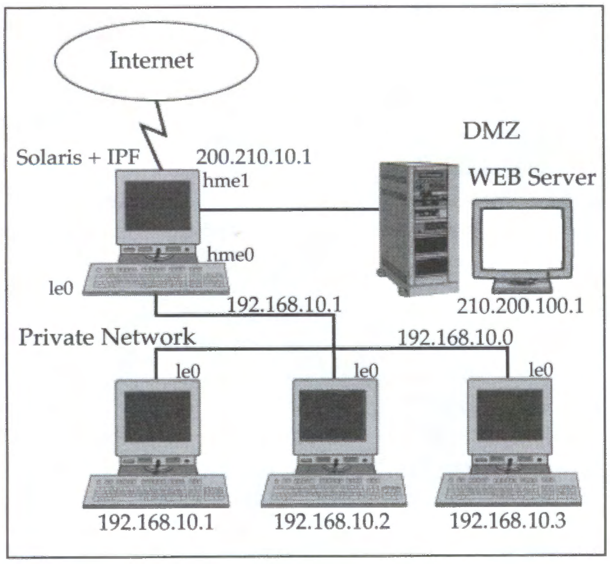


Figure A: *This is a simple network configuration appropriate for a small office.*

ZIFF-DAVIS a SOFTBANK company

requests. In this case, you must create a proxy application for each networked service (one for ftp, another for telnet, another for http, and so forth). Even though a proxy firewall is application aware, it can provide more security than any other type of firewall.

In this article, we'll show you how to install, configure, and run a freely available firewall—Internet Packet Filter (IPF). IPF is a hybrid firewall, which is a dynamic packet filter with some proxy capabilities. Written by Darren Reed, IPF is capable of selectively filtering any protocol listed in the /etc/protocols file. You load IPF into the Solaris kernel between the data link (the actual NIC) and the network (for example, IP) layers (layers 2 and 3 of OSI model). By inspecting at this layer, IPF intercepts and inspects all inbound and outbound packets on all interfaces.

Preparing the system

In order to function as a firewall, your machine should have at least two network interfaces in addition to lo0 (loopback). If you've met these requirements, begin system preparation by installing only those packages that are absolutely required by the OS. Sun allows you only to install as a minimum the core meta-cluster. Because this cluster contains packages that aren't really needed for a properly secured system, delete the following:

- Files that present specific security concerns, such as snoop, finger, rlogin, rdist, etc.
- Files that are part of a package that's deemed unnecessary for a firewall, such as NIS, UUCP, and the Solaris Desktop/CDE environment.
- Files that will be replaced by a more secure version of that service, such as in.telnetd and in.ftpd (which will be replaced with SSH) or aren't necessary for a secure build (for example, nfs.client, nfs.server, rpc, autofs, sendmail, in.routd, etc.).

Now, install the latest recommended patch cluster and Y2K patch cluster and comment out all unnecessary services in the /etc/inetd.conf file. Next, install the following third-party software tools, which are required to complete a secure build: SSH, S3, Tripwire, and fix-mode. Ensure that all interfaces on the machine have unique MAC addresses, including the virtual address, by executing the following command:

```
#eeprom local-mac-address?=true
```

Ensure that interface load balancing is not enabled (Solaris 2.6 only). All outgoing data from the same source must have the same source IP address:

```
#/usr/sbin/ndd -set
➥ /dev/ip ip_enable_group_ifs 0
```

Now, enable IP forwarding with the following command:

```
#/usr/sbin/ndd -set ip_forwarding 1
```

Optionally, you can prepare two separate Solaris boxes to act as one firewall. Then, install a High Availability system between these boxes and run in asymmetric mode.

Installing IPF

Now you need to install the IPF. You'll find the latest version of the IPF source code (currently version 3.3.6) at:

ftp://coombs.anu.edu.au/pub/net/kernel

Once you've downloaded the IPF, it's easy to install. To do so, simply type the following command:

```
#make solaris
```

This will work for both Solaris 2.3-6 and Solaris 2.4-6x86. Just make sure to use Sun's native make utility—/usr/ccs/bin/make—as the GNU make won't work!

Once you've successfully compiled the IP Filter, change the directory to SunOS5 and type:

```
#make package
```

This generates a packaged version of the IPF and you'll be asked by a pkgadd utility if you would like to install it. If you decide not to install IPF at this point, you can do it later on by issuing the following command:

```
Pkgadd -d .
```

If you decide to proceed with an installation, IPF installs the package and loads the IPF kernel module from the /usr/kernel/drv directory. In addition to the kernel modules, IPF installs files into the /usr/sbin, /opt/ipf, /etc/opt/ipf, and /etc/init.d directories.

To check if the installation completed, issue the following commands:

```
#modinfo | grep ipf
#pkginfo  ipf
```

If you see the information about IP Filter, you're all set.

Creating IPF filters

We'll base our examples on the simple network topology depicted in Figure A. The /etc/opt/ipf/ipf.config is a configuration file in which we'll put all our filter rules. The format of this file is simple; there's only one rule per line, and the pound sign (#) indicates a comment. The rules syntax is simple and easy to understand. Suppose we'd like to allow HTTP connections from the Internet only to our Web server with an IP address of 210.200.100.1. In this case, the ipc.config file will look like this:

```
pass in log quick on hme1 from any
➥ to 210.200.100.1/32 port = 80

block in log quick on hme1 from
➥ any to any port = 80
```

The first line is a rule that says "Pass and log all incoming traffic (`in`) that goes through the hme1 interface (`on hme1`) from any source to the target 210.200.100.1 (`/32` is a netmask), and that's addressed to the HTTP (`port = 80`)." The keyword `quick` tells IPF to take this action immediately and not to bother checking the rest of the filter table. To enable these rules, type:

```
#/etc/init.d/ipfboot stop
#/etc/init.d/ipfboot start
```

To allow only established connections to pass through, we could modify our ipf.conf file as follows:

```
block in on hme1
➥ pass out quick on hme1 proto tcp/udp
➥ from 210.200.100.0/24 to any
➥ keep state

pass out quick on hme1 proto icmp
➥ from 210.200.100.0/24 to any
➥ keep state

pass in quick on hme1 proto tcp
➥ from any to 210.200.100.0/24
➥ port = 80
```

The workings of the keep state rule set are much like the workings of the saying "do not speak until spoken to." Communication into the firewall is just not permitted (except on port 80). See Listing A for more examples of firewall rules that are worth adding into the configuration file.

As you can see from all these examples, IPF has many different options and various rule-setting syntaxes. For detailed information about IPF's options, consult IPF(5)'s man pages. You can also examine the example's rules found in /opt/ipf/example.

IPF and NAT

NAT (Network Address Translation) converts all the hidden source addresses and port numbers behind the firewall to the IP addresses of the firewall running NAT. Our private network in Figure A uses non-routable IP addresses. These ranges are defined by the Network Information Center for private use only (RFC1869—"Address Allocation for Private Internets"). So NAT effectively allows systems on this network to share a single, registered IP address on the Internet. This allows all of the machines behind the firewall to use Internet services such as http, ftp, telnet, and email.

Listing A: *Examples of firewall rules*

```
# Block packets that are too short
# to contain a complete header.
# Block packets with IP options:
block in log quick from any to any with short
block in log quick from any to any with ipopts

# Block broadcasts:
block in quick from any to 0.0.0.255/0x000000ff

# Block Anti-spoofing attacks:
block in log quick on hme1 from 127.0.0.0/8 to any
block in log quick on hme1 from 10.0.0.0/8 to any
block in log quick on hme1 from 192.168.0.0/16 to any
block in log quick on hme1 from 172.16.0.0/12 to any
```

In order to enable NAT, we have to create a file called /etc/opt/ipf/nat.conf. By adding the following line into the nat.conf file for our private network, we will say

```
map hme1 192.168.10.0/24 ->
➥ 200.210.100.1/32
```

Whenever a packet goes across the hme1 interface with a source address matching the 192.168.10.0/24 it will be rewritten within the IP stack such that its source address is 200.210.100.1. Then the packet will be sent to its original destination. The system also keeps a list of which translated connections are in progress so that it can perform the reverse and remap the response (which will be directed to 200.210.100.1) to the internal host that really generated the packet.

If you use a dynamically assigned IP addresses, you have to rewrite this rule:

```
map hme1 192.168.10.0/24 -> 0/32
```

In this case you'll have to run `ipf -y` to refresh the address if you get disconnected. But what happens to the source port when the mapping occurs? The packet's source port is changed to the first available port on the firewall. If you don't desire this behavior, and wish to limit the amount of ports that the NAT system can consume, we can add the portmap keyword:

```
map hme1 192.168.1.0/24 -> 0/32
➥ portmap tcp/udp 10000:30000
```

Now the NAT translates all connections (which can be TCP, UDP, or TCP/UDP) into the port range of 10000 to 30000. For more information about NAT, refer to the manual pages or the "IP Filter Based Firewalls HOWTO" found at www.obfuscation.org/ipf/ipf-howto.txt.

IPF, TCP Wrapper, and Nmap

If you install IPF on a single host with one network card, you can still benefit from the services that IPF provides. If you're a longtime reader of *Inside Solaris*, you should be familiar with TCP Wrapper. Now let's see what advantages IPF has over TCP Wrapper.

First, TCP Wrapper cannot protect services that start at boot time and run until system shutdown, like sendmail and httpd. IPF doesn't have this limitation. In addition to TCP, it can also block UDP, ICMP, and other protocols. Assume that we installed IPF on 192.168.10.1 in order to protect this host from any other hosts on a private network, as shown in Figure A. Let's see how we can block various TCP and UDP services:

```
# Allow telnet to pass from
# 192.168.10.3 only:
pass in quick on le0 proto
➥ tcp from 192.168.10.3
➥ to 192.168.10.1 port = telnet
➥ keep state

pass in log quick on le0
➥ proto tcp from 192.168.10.3
➥ port > 1023 to 192.168.10.1
➥ port = telnet keep state

block in log quick on le0 proto tcp
➥ from any to any port = telnet
```

If we would like to stop rlogin and rsh services, we have to add the following rules:

```
block in proto tcp from any
➥ to any port 512 >< 514
```

This filter will prevent any incoming TCP connection to ports 513 (rlogin) and 514 (rsh). To block and log any inbound UDP packets that are going to the NFS port 2049, you can write the following rule:

```
block in log on le0 proto udp
➥ from any to any port = 2049
```

In addition, IPF can fool nmap and other Internet scanners. To misguide the attacker into believing that there are no services to break into, we can create our own TCP response. When a service isn't running on a UNIX system, it normally replies with an RST (Reset) packet. When blocking a TCP packet, IPF can actually return an RST to the origin by using the `return-rst` keyword:

```
block return-rst in log quick
➥ from any to 192.168.10.1 proto
➥ tcp port = telnet
block in log quick on le0
pass in all
```

If somebody sends a packet to a UDP port, we can use the `return-icmp` keyword to send a reply:

```
block return-icmp (port-unr) in log
➥ quick on le0 proto udp from any to
➥ 192.168.10.1/32 port = 111
```

In this case, a `port-unreachable` message will return as if no service is listening on the port in question.

Unfortunately, if you don't block a service completely, a SYN type of attack cannot be defeated with IPF. For this reason, you might want to log all initial SYN packets for further investigation:

```
log in on le0 proto tcp from
➥ any to any flags S/SA
```

Tools

IPF comes with various tools that help you perform comprehensive monitoring and debugging. Because of the limited space of this article, we won't go into details about all of them.

First, we'd like to mention the ipmon utility. Ipmon watches the packet log as created with the log keyword in our rules. Running in the background with the -o option, ipmon will send packet information through syslogd. The default facility for this is local0. If you add an entry into the /etc/syslog.conf file (for example, local0.info /var/adm/ipf.log), you can use the ipf.log file as a pattern-matching source for any host-based intrusion detection system.

If you decide to watch events as they happen in real-time, run ipmon with -o I options. Also, ipmon lets you look at the NAT table in action (ipmon -o N) and the state table (ipmon –o S).

To see a snapshot of how IPF is doing, use the ipfstat utility. You'll see a statistic of how many packets have been passed or blocked, if they were logged or not, how many state entries have been made, and so on. It's also capable of showing your current rule list.

With ipftest, you can test your rule file. It reads in a filter file and then applies sample IP packets to it. This allows for testing of the filter list and examination of how a packet is passed along through it. And mkfilters will suggest that you use a set of filter rules to add routes to back this up. For more information about these and other utilities, consult IPF's man pages.

Further Reading:

Since it isn't possible to cover all firewall/IP filter aspects in one article, an excellent book about firewalls in general is *Firewalls and Internet Security: Repelling the Wily Hacker*, by William R. Cheswick and Steven M. Bellovin. One of the best references for how to secure a Solaris system is *Securing Solaris: Step-by-Step*, which you can find at www.sansstore.org. Further, you can find a list of free and commercial firewall products in *Maximum Security: A Hacker's Guide to Protecting Your Internet Site and Network*. And of course, for more information about IPF, you can check the *IP Filter Based Firewalls HOWTO*.

Secure intranet file transfers

by Paul A. Watters

As businesses rapidly become reliant on data transfers across the Internet using TCP/IP, stories of leaked documents and stolen data are becoming increasingly common. In this context, the term intranet means any kind of data that's exchanged physically outside a single site or office, to employees or remote units, rather than just referring to a Web-based organizational tool.

In addition to sending memos and providing services for internal job scheduling and messaging, intranet technology typically involves large numbers of file transfers across many different locations. These transfers include static Web data to remote proxy servers from a central information service, as well as shared uploads of non-critical data that needs to be accessed by multiple users. Developing technology that satisfies the multiple constraints of making data available globally, as well as encouraging employees to make use of the technology by ensuring that it's easy to use, has proven difficult, and has exposed security flaws in the designation of trusted systems.

Passwords

Many of us will have issued memos to administrative departments and offices to change all passwords because a password sniffer has been detected on the network, for example, or because a cracker has breached a firewall from an unauthorized host. Users are often unaware of the implications of transmitting passwords in plain text across the Internet. A common opinion is that since Solaris passwords are encrypted, they must be safe.

Although encrypted passwords are relatively safe even if the password file (/etc/passwd) is retrieved by a thief, the actual characters typed across a serial line or ethernet can all be potentially detected. In addition, globally networked users often want access to large amounts of disk space to exchange non-critical data with colleagues in remote locations, without having to set up multiple user accounts for each new file transferred.

Anonymous ftp is a potential minefield, although measures can be implemented to ensure that the host system isn't compromised, or that it cannot be misused (for example, by imposing user quotas). In this article, we'll examine some methods, including alternatives to ftp and rcp, to improve the security of file transfers within intranets.

Identification and authentication

Rcp is the standard Solaris command to copy files and directories to file systems on other local servers. It's often used to mirror filesystems, and has a very simple command syntax. You can use an rcp command embedded in a script to specify a source file or directory, a target host and a target user, and an optional top-level directory into which to copy the source files or directories.

If you don't specify a target directory, the files or directories will copy into the home directory of the remote user. Likewise, if the user on the source machine has the same username as the target machine, you can omit this information on the command line.

Drawbacks

Rcp has one major drawback, despite its convenience: passwords and the contents of remote copy sessions are transmitted in plaintext across the Internet. If a company office in San Francisco wants to back up its daily work by using rcp to a data warehouse in Dallas, then the username and password are available for sniffing by any privileged user on a computer located between the two sites.

In addition, if the company data being transmitted is sensitive, it's quite possible that a competitor might attempt to obtain commercial advantage by intercepting this data. A related problem could involve the remote copying of company data that's fake from a source host with a valid username and password.

Improve the security of remote copying

Fortunately, these two issues of identification and authentication have been addressed by a number of commercial products that improve the security of remote copying. The F-Secure suite by Data Fellows found at www. datafellows.com, for example, contains the scp (secure copy) utility, which both encrypts the exchange of username and password data between a local and remote host, as well as the data to be transmitted. A command to securely copy the file topsecret.txt between the local server and a remote server called `remotehost` would look like the following:

```
unix% scp topsecret.txt backup@remotehost:
backup@host's password:
topsecret.txt                     |
   1 KB |   1.3 kB/s | ETA: 00:00:00 | 100%
```

If the local user hasn't attempted to make a connection to the remote computer and user account previously, he'll be prompted to accept the session key returned by the connection request from the client to the server. If he accepts the session key, the session can be approved. In this example, the user account *backup* on the target machine has been accessed from this machine previously.

Using this kind of secure client provides an improved mechanism for copying local files remotely, ensuring that clear text passwords are *never* transmitted across the Internet. A formal authentication mechanism based on the existence of remote and local certificates can also be instituted with the F-Secure utilities, so passwords are no longer required. Access is only granted if a source machine and user have the appropriate certificate. This allows you to securely automate backup scripts for remote sites.

Enforcing quotas

If the overhead caused by encrypting the contents of each session causes significant delays in network transmission, you can use other methods to improve security on intranet file transfers, if the data is non-critical and access to disk areas isn't password protected. The main concern about an open area like this is denial of service. One simple remedy is to enforce disk quotas on all public file systems.

Although many organizations have a policy of not enforcing disk quotas for fear of discouraging development activity, a successful denial-of-service attack comes in the form of maliciously filling world- or group-writeable disk slices with large files. If a server has an anonymous ftp server with an incoming area, or shares a globally-writeable NFS share to other local hosts, these slices can be filled to capacity, preventing system services from being carried out.

For example, imagine that a server has a 9-GB disk, which has a single partitioned slice mounted as /public. The directory /public/ftp/incoming is writeable by the ftp user, so a malicious user can easily put large files in the incoming directory, thereby preventing any legitimate users from putting their own files into your file system.

If the organization has a policy of allowing only files less than 512 MB, for example, you can add a find command as a cron job every hour to delete files that don't meet the requirements of the policy. In any case, the fact that the disk is full will only be a temporary inconvenience.

However, imagine that the /var partition has been filling to capacity recently, and the administrator decides to create a symbolic link from /var/mail to /public/mail. Of course, he sets the appropriate permissions on the /public/mail subdirectories so that only legitimate users can read their email. However, if a malicious user puts 9 MB of files less than the 512-MB limit into the incoming directory, the entire disk would fill up, and users would not receive any email for some time.

Worse still, if the attack isn't detected in less than four days (for example, over a holiday weekend), then valuable email will be bounced to the sender, and lost forever to the user. For businesses that rely on encrypted email orders to sell goods and services on the Internet, each lost message is associated with a tangible loss.

Limit the amount of disk space available to the anonymous ftp user

Apart from the obvious solution of never creating symbolic links from system areas to public areas, and physically preserving the /var partition, a better solution would be to place a total limit on the amount of disk space available to the anonymous ftp user. Although there's an administrative overhead associated with quotas, you can minimize this by only setting quotas for the filesystems to which the anonymous ftp user has access.

Continuing with our example, edit the /etc/vfstab file as root, and add the rq flag to the mount options field for the /public file system. After changing the directory to /public, create a file called *quotas*, and set permissions on the file to be read and write for root only:

```
unix% cd /public
unix% touch quotas
unix% chmod u+rw quotas
```

Next, you can create quotas for the /public filesystem for the anonymous ftp user by using the command:

```
unix% edquota ftp
```

The allocation of disk space on the /public filesystem is made on the basis of the number of inodes and 1K blocks made available to the user. You can check your work by viewing the quota on all the filesystems available to the anonymous ftp user by typing:

```
unix% quota -v ftp
```

If the quotas are as expected, the quota system for /public can be activated by issuing the command:

```
unix% quotaon /public
```

Quota sizes are tunable, so if users complain that insufficient disk space is available to meet their needs, the quota for /public can be increased by using the edquota command.

Further Reading:

An excellent source of information about Solaris security is the Solaris Security FAQ, maintained by Peter Galvin, at the Sun World site located at www.sunworld.com.

Inprise releases JBuilder 3 Solaris Edition

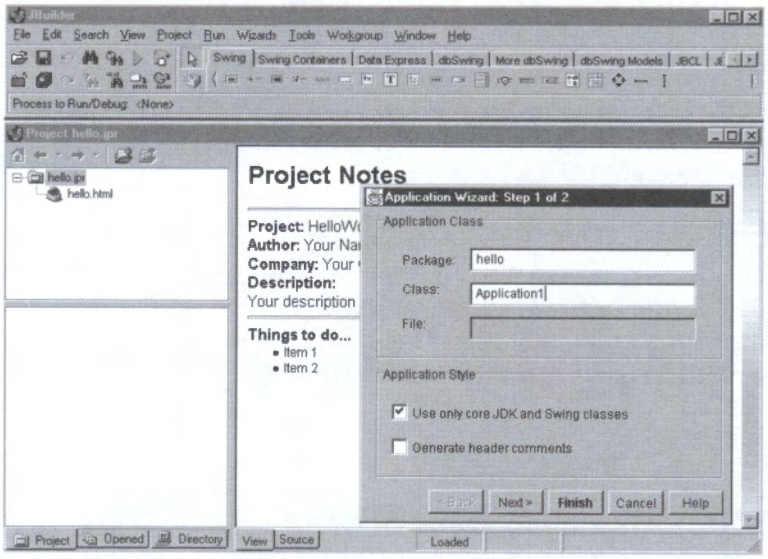
by Clayton E. Crooks II

Inprise has released the latest version of their rapid application development tool, JBuilder 3, for creating business, database, and distributed applications based on the Java 2 platform. The product provides Solaris developers with the visual development tools that have made JBuilder a leading enterprise Java-development tool on the Windows platform. The product supports J2EE (Java 2 Enterprise Edition) so programmers can rapidly deliver reliable and scalable applications written entirely in the Java language.

What can JBuilder do for you?

First, JBuilder 3 provides reusable components and visual tools for rapidly creating platform-independent applications for the Java 2 platform, as well as wizards for creating reusable JavaBean technology. In addition, JBuilder 3 supports CORBA development for dramatically reducing the time and effort required to deliver high-availability CORBA clients and servers.

CORBA (Common Object Request Broker Architecture) is an emerging open-distributed, object-computing infrastructure which the Object Management Group (**www.omg.org**) is standardizing. CORBA automates many common network programming tasks such as object registration, location, and activation; request demultiplexing; framing and error-handling; parameter marshalling and demarshalling; and operation dispatching.


JBuilder
File Edit Search View Project Run Wizards Tools Workgroup Window Help
Swing | Swing Containers | Data Express | dbSwing | More dbSwing | dbSwing Models | JBCL
Project hello.jpr
hello.jpr
hello.html
< Back Next > Finish Cancel Help
Project Opened Directory View Source Loaded

Process to Run/Debug <None>

Project Notes

Project: HelloW
Author: Your Na
Company: Your
Description:
Your description

Things to do...
- Item 1
- Item 2

Application Wizard: Step 1 of 2

Application Class

Package: hello
Class: Application1
File:

Application Style

[✓] Use only core JDK and Swing classes
[ ] Generate header comments

Figure A: *This is the workspace in JBuilder's integrated development environment.*

JBuilder for Linux, Windows, and Solaris

Inprise is also releasing JBuilder for Linux in the near future, and, along with the Solaris version, they will be the first Java development environments that are written entirely in the Java language. These products will also be the first Java development tools that will allow programmers to develop within the Linux or Solaris environments and deploy to multiple platforms.

JBuilder currently runs under Windows 95, 98, and NT/2000. The Linux version of JBuilder is due out by the middle of 2000. Inprise is also considering porting JBuilder to more flavors of UNIX in the future.

The Solaris version of JBuilder looks and feels much the same as the very popular Windows version, but has more Java features. This new version is Java 2 Enterprise Edition (J2EE) compliant, putting it a step ahead of its Windows counterpart. Under Windows, JBuilder 3 is compliant with Java 2 Standard Edition, a version of Java that's less oriented toward Java server applications.

JBuilder 3 is noted for its IDE that includes the ability to debug applications running on remote servers and workstations. Further, an improved editor with support for emacs (an open source UNIX development editor) and Common User Access key bindings gives terminals access to Java applications. IDE's interface is shown in Figure A.

You'll find that JBuilder 3 for Solaris provides an extensive set of tools that will be a good fit for new or experienced Java developers. It's also a worthwhile upgrade for current JBuilder customers and a solid solution for sites looking to obtain Java development tools. You can find JBuilder for Solaris at **www.inprise.com/jbuilder/solaris/**. The price ranges from $99.95 for the standard edition to $2,499.00 for the full Enterprise version.

Easy device driver development

by Clayton E. Crooks II

WinDriver for Solaris is a device driver development tool kit with a user mode API. It allows you to access your hardware directly from your application without having to learn the Solaris internals and without having to master and use the Device Driver Interface /Driver Kernel Interface (DDI/DKI).

WinDriver supports the PCI, ISA, ISA PnP, and EISA buses on both Sparc and Intel platforms. The same driver code you write will run on both platforms.

The Solaris version of WinDriver features full compatibility to all other versions of WinDriver. This is significant in that it offers full portability between Solaris, Linux, Windows NT/2000, 95/98, NT Embedded, CE, and OS/2, without changing a single line of code.

Writing a device driver

Writing a device driver using WinDriver for Solaris is an easy and concise process. To begin, download and install WinDriver for Solaris from www.krftech.com/dnload.html.

Next, download and install WinDriver for Windows from the same Web site. Now, plug your hardware into a Windows machine. Run the Driver Wizard and choose your hardware. You may view or define its resources at this time.

Then, use the Graphical Wizard to verify that your hardware works as expected, by reading from and writing to your hardware. Next, use the wizard to automatically generate your driver code. (The wizard also generates make files for Solaris and Windows.)

Finally, move your hardware and the generated code back to your Solaris machine and recompile. Your driver will now work on Solaris (and is also portable to all other supported OSs).

Don't have Windows?

If you don't have access to a Windows machine, you can alter one of the many examples you obtained with the download. All samples compile and run on any supported OS. They are a great way to get started on a driver and may provide most of the needed work.

Support

The Solaris version includes support for interrupt handling, I/O and memory handling, DMA support (in x86), optimal performance module, and all other features of the Windows version. WinDriver for Solaris also includes enhanced support for PLX, Altera, QuickLogic, and other PCI chip-set vendors. This includes a robust, ready-made driver, compatible to all operating systems, and an extensive API with which you can perform any operation supported by these chip sets (DMA, EEPROM, Interrupts, and so on).

If your job requires the development of device drivers, WinDriver is an easy-to-use option you shouldn't overlook. In most situations, WinDriver will save many hours of hard kernel development for each operating system and project.

The Common Desktop Environment

by Edgar Danielyan

All the differences between various UNIX implementations have contributed to UNIX's reputation as a powerful but problematic operating system. In response to these issues, a group of vendors formed the Common Open Software Environment (COSE) to create standard APIs for UNIX. One of these is the Common Desktop Environment (CDE), the latest version of which is bundled with Solaris 7 and the upcoming Solaris 8. In this article, we'll take a look at the features of the CDE and how to make the best use of them.

Features and benefits

The CDE has many great features and benefits. The most important are:

- The unified and consistent look and feel throughout all supported platforms
- A rich set of bundled, fully integrated applications, such as File Manager, Desktop Manager, Mail Application, Text Editor, and various system management applications
- The third-party software that's available

The CDE has all the features you would expect from a modern, graphical, user-interface system. The Solaris implementation of the CDE even provides full support for OpenStep, Sun's object-oriented programming environment, and OpenWindows. The same user can use CDE, OpenStep, and OpenWindows simultaneously on the same machine.

The graphical login daemon (dtlogin) for Solaris provides a choice of login options, and the user can select either the Common Desktop Environment or OpenWindows. The CDE is pretty configurable, and aside from being able to change colors, window positions, and the like, users are able to communicate with the CDE applications via ToolTalk, which is an application messaging protocol and system used by all CDE applications.

Common Desktop

One of the concepts of the CDE is the Desktop. The Desktop contains the front panel and multiple workspaces (virtual desktops). By default, the front panel shown in Figure A, which is located at the bottom of the screen, displays icons and has roll-up menus. It contains the most frequently used applications, including the clock, calendar, load indicator, Exit button, and anything else the user decides to place there.

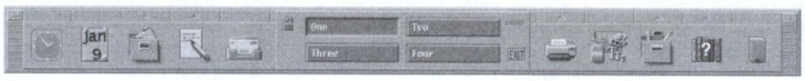


Figure A: *The front panel in CDE displays icons and roll-up menus.*

One method to place applications on the Desktop is to drag and drop the application from File Manager, for example. The CDE supports file action binding and typing, just like Microsoft Windows. Further, the CDE comes with a number of pre-configured actions, and it's possible to add new file types and actions easily.

From the programming viewpoint, the CDE is largely based on Motif (in fact, the CDE Window Manager is Motif's MWM). However there are a number of extensions, mainly where the original Motif didn't provide required functionality. Also, there are some hooks to permit native OpenLook applications to run without any modification. Included with the CDE are the utilities and applications found in Table A.

All these applications support drag and drop as well as ToolTalk. For additional convenience, some daemons, such as the vold (Volume Manager) and lpd (Print Manager), have the CDE front-ends. The interface is

Table A: *Utilities and applications included with the CDE*

Application manager
Audio Tool
Calculator
Calendar
Icon editor
Image viewer
Clock
Snapshot
Text editor
Text note
Voice note
Address manager
File manager (with Find File and trash)
Mailer (MIME compliant)
Admintool
Power manager
Print manager
Process manager
Style manager
Desktop Korn Shell (dtksh)
Terminal (VT220)

intuitive, but in case help is needed, all bundled applications have an extensive Help facility.

Conclusion

For the foreseeable future, the Common Desktop Environment will be the GUI of choice for Solaris and other commercial UNIX systems. Compared with OpenWindows, the CDE is a little heavier, but that's balanced by the CDE's open interoperable user interface, new features, and flexibility of configuration. But of course, you don't have to give up your beloved xterm and vi!

Installing Solaris x86

by Paul A. Watters

Solaris for the x86 architecture is now reaching maturity. It's being released simultaneously with the Sparc version of Solaris, including the new Solaris 8, which is presently in Early Access release. Most organizations are currently deploying Solaris 7 across the enterprise, although many of the benefits for data centers in Solaris 8, including seamless cluster computing, are attractive.

In this article, we'll walk through the early stages of Solaris 7 x86, which differs substantially from those typically followed for the Sparc architecture. However, users of Solaris for the Sparc architecture will find the latter stages of the installation process, such as the configuration of disk slices, reassuringly the same as those they are used to using. We'll also outline some of the pitfalls of installation and some commonly encountered problems in device configuration.

Why install Solaris x86?

With the release of the Ultra 5 and Ultra 10 servers from Sun, which are competitively priced with respect to high-end x86 systems, many users will ask why they should consider the move to the x86 platform. There are two important reasons.

First, many organizations have a vast supply of x86-based machines that could benefit from a change to the Solaris operating system. These systems are often used as servers for workgroups, providing email and Internet gateway services, using one of the many operating systems available for the x86 architecture. However, the security and user management features of Solaris are well-known in the server world, and the implementation of Solaris x86 has been welcomed by many department-level administrators.

The second key reason for considering the x86 platform is that there's a much larger variety of hardware available for the x86 platform than for the Sparc architecture. For Solaris to successfully leap to the desktop and small office/home office market, it needs to support the many printers, scanners, and ISA- and PCI-based peripherals that are available from third-party hardware vendors.

Having stated the advantages of x86, it's hard to see how x86 would ever replace the E10000, which is suited to mission-critical, high-performance applications. However, if the many x86 servers and high-end E10000 servers were running the same operating system, data sharing and site configuration could be easily streamlined, reducing maintenance and administration costs.

Selecting hardware

The golden rule for a trouble-free Solaris installation is to either follow the tested system configurations listed in the Hardware Compatibility List (HCL), and/or only use the hardware for which device drivers are available. Although it's quite possible that a particular system may work if it uses standard components (for example, a standard PCI motherboard), some vendor-specific innovations may not be directly or indirectly supported.

Of course, if there's a specific hardware device that you really need to support, but for which there's currently no device driver available, it's always possible that your harware vendor will provide resources for writing one, if the demand is strong. This appears to be the case for RAID controllers, for example, which are commonly used in servers for enterprise computing.

The Solaris 7 x86 distribution consists of a single floppy disk, containing the Device Configuration Assistant (DCA), and a CD-ROM, containing the Solaris platform. The dual-media (floppy and CD-ROM) approach is different from the standard installation on Sparc systems, which can be booted from a CD-ROM alone. The latter stages of x86 installation are identical to that for the Sparc, so we'll concentrate on the steps involved in configuring devices and booting the Solaris kernel for the first time.

Pre-install preparation

To install Solaris, insert the Device Configuration disk into the floppy disk drive. Enter the system's BIOS configuration utility (usually by pressing [Delete] after the memory check). Now, select the booting device order to include the A: drive first, as many server systems are configured to boot directly from the hard drive to reduce boot time (usually the C: drive). An acceptable drive order would be A:, C:. Save the changes to the system settings and exit. The system should now restart and boot into Solaris Device Configuration Assistant.

Device configuration

The first program to load is the Solaris Boot Sector (version 1), followed by the Solaris for x86 FCS DCB. A mini-kernel (/solaris/boot.bin) is then loaded, after which the SunOS Secondary Boot (3.00) initializes. If all goes well, the system is then able to run the configuration assistant.

A potential problem that can occur at this point is that the floppy disk loads the Boot Sector program, but fails to boot the FCS DCB, displaying an error message stating it doesn't recognize the boot device. This is usually caused by non-standard hardware, such as a Zip disk or an LS-120 removable disk, being configured as the boot device. Configure a standard floppy disk drive as the boot device in BIOS, if one is attached to the system, or borrow one from another system.

The Device Configuration Assistant offers the following three alternatives for configuring the Solaris installation:

- Perform a full scan to identify all system hardware
- Diagnose possible full-scan failures
- Add new or updated device drivers

As new hardware becomes available, updated device drivers will be available from Sun or hardware vendors. In addition, you can download updated Device Configuration Assistant disk images from Solaris's Web site at www.sun.com/Solaris. Note that the mouse cannot be used during configuration.

Full scan configuration

If the system performs a full scan, the DCA attempts to determine the bus type, and then gathers hardware configuration data. Most modern bus types (for example, PCI and SCSI) are supported in Solaris x86, as well as legacy bus types like ISA.

After building a driver list, the DCA then detects system hardware using an exhaustive search of the most common drivers supplied on the floppy disk. A progress bar is displayed, along with the names of candidate devices being scanned for. In most cases, if hardware is listed on the Hardware Compatibility List (HCL), the full scan will detect the devices appropriately. However, if there's an observed resource conflict, the affected devices are marked with an exclamation point (!).

In order to use these devices, you must resolve the resource conflicts. You can usually achieve this by modifying hardware settings on one of the devices for the interrupt request level (IRQ), direct memory access channel (DMA) number, or base memory address. If these cannot be set by jumpers on the device board, it should be possible to configure them using the BIOS configuration menu.

If devices connected to your system don't appear in the list of identified devices, select *Device Tasks* to view or edit the list of available devices, and/or save a configuration specific for your hardware. This enables rapid configuration and installation of Solaris using a standard operating environment (SOE) for many systems concurrently, which share the same hardware setup. This is a common approach to installation in large organizations that are converting their x86-based server farms to Solaris from another operating system.

The list of identified devices on our NEC Versa laptop were identified automatically by the Device Configuration Assistant. They also appeared in the identified device list without any resource conflicts on both the PCI and ISA buses:

```
ISA: Floppy disk controller
ISA: Motherboard
ISA: PS/2 mouse
ISA: PnP bios: 16550-compatible serial
     controller
ISA: PnP bios: Audio device
ISA: PnP bios: Parallel port
ISA: System keyboard (US-English)
PCI: Bus Mastering IDE controller
PCI: VGA compatible controller
```

Specific Scan and new drivers

The Specific Scan option allows us to manually select all of the devices that we know are attached to our system. This is a much more comprehensive setup process than the Full Scan method, but may be useful if devices aren't detected automatically.

If the Adding New Or Updated Device Drivers option is selected, then the Device Configuration Assistant prompts the user to insert a supplemental disk that contains the new drivers. This is an important step if new hardware is required as part of the installation process (for example, a CD-ROM driver).

Booting Solaris

After configuration, a number of drivers required for installation are loaded (for example, com.bef and ata.bef). The next step is to select the installation media. Although first-time users will almost always install from a CD-ROM, it's also possible to load a local hard drive with the contents of the CD-ROM, and use that as the installation medium. This is an option if the CD-ROM attached to the system isn't supported by Solaris, but is supported by another operating system. Alternatively, it's also possible to install across a network, but network settings for the new system (for example, hostname, IP address, and DNS server) must be obtained prior to installation.

For the non-networked NEC Versa whose configuration is given, there are two options: CD-ROM or hard drive. Once the CD-ROM option is selected, the installation procedure begins with a boot of the Solaris kernel, and the process is similar to that for a Sparc installation.

After displaying the boot device parameters

```
pci@0,0\pci-ide@14\ide@1\sd@0,0:a
```

users are prompted to begin one of three types of installation:

```
1. Solaris Interactive;
2. Custom JumpStart;
3. Solaris WebStart;
```

Unless a customized JumpStart or WebStart configuration had been created previously, most users will want to enter Solaris Interactive boot option.

At this point, the kernel is booted, and the usual messages are displayed:

```
Booting kernel/unix
SunOS 5.7 Version Generic [UNIX¿ System V Release
   4.0]
Copyright ⌐ 1983-1998, Sun Microsystems Inc.

Configuring devices...
```

After the device configuration created by the Device Configuration Assistant is loaded, the installation process begins with the kdmconfig program, which is the graphical user interface (GUI) installation program. After entering the configuration of your monitor and video card, you should be able to begin the installation process using OpenWindows.

Further Reading:

The Solaris Web site, www.sun.com/solaris, contains excellent information about the x86 release of Solaris, and has many tips and pointers regarding configuration and installation. Similarly, the USENET forum alt.solaris.x86 and the e-groups list solarisintel, found at www.egroups.com/, both have extensive coverage of the x86 release.

For further information regarding the configuration of specific devices, such as ethernet cards, display settings, sound cards, RAID devices, and SCSI controllers, see the *Solaris 7 (Intel Platform Edition) Device Configuration Guide.* This guide is typically supplied on a separate CD-ROM with the Solaris distribution, but is also available in HTML and PDF formats at the Sun Documentation Server (docs.sun.com).

Recordable CD-ROMs for Solaris

I work in a multimedia production company, and many of my users want to be able to record CD-ROMs, containing both music and data, on a day-to-day basis. Does Solaris support recordable CD-ROMs, or do I have to buy a PC?

The good news is that Solaris does support both reading and writing CD-ROMs. The technical ability to support any SCSI-based device is a given for the operating system, but a potentially limiting factor for non-standard hardware is usually finding software to adequately support it. Fortunately, both commercial and freeware versions of CD-recording software are available for Solaris.

One of the most popular freeware tools is cdrecord, by Jörg Schilling, which is available at **ftp.fokus.gmd.de/pub/unix/cdrecord/**. This application supports both Solaris 1.x (SunOS 4.1.3 and later) if you have the SCSI general driver scg. Solaris 2.3 and later are supported through the use of the real-time scheduler. It even runs on the Solaris x86 platform, and supports both music and data recordings. Although it's based on a command-line interface, it has many of the features of expensive GUI systems, and includes the following capabilities:

- Simulate a recording for test purposes (dummy option)
- Use a single CD-ROM for multiple recording sessions (multi option)
- Manually fix the disk if you want to view data from an open session on a normal CD-ROM (fix option)
- Set the recording speed factor (speed option)

Popular commercial alternatives to cdrecord include GEAR for UNIX, found at **www.gearcdr.com/html/products/gear/unix/index.html**, and Creative Digital Research's CDR Publisher, **www.cdr1.com/**, which is available through Sun's Catalyst program. For more general information about the CD-ROM recording process, see Andy McFadden's very comprehensive FAQ at **www.fadden.com/cdrfaq/**.

Solaris Device Configuration Assistant

I recently had a hard disk failure on my Solaris 7 x86 system, but when I tried to use the boot floppy disk, it no longer worked. Do I have to order a new one?

Fortunately, the Solaris Device Configuration Assistant (boot.zip) is available for download at **soldc.sun.com/support/drivers/boot.html**, as part of the Solaris Developer Connection. There are also driver updates available at the same location. You can then copy the new boot file to a floppy disk using dd (on another Solaris machine), or by downloading the Windows version of dd. After rebooting the machine, insert the floppy disk, and you should be able to reconfigure your system.

Turning off diagnostic mode

I recently received my Sparc 2 back from servicing by a third-party, and I've found that I need to type b vmunix *to boot the system at the boot prompt. This is getting annoying, as there's a long delay after the memory test is complete. Why is this happening?*

Most likely, the `diag-switch?` variable in the OpenBoot monitor has been set to true by the service people for diagnostics. In this mode, the Sparcstation will perform a number of diagnostic tests, and identify any potential hardware faults. In addition, it will set the default boot device to *net*, which will fail if your machine isn't configured for network booting. Setting the diagnostic mode to true is useful for checking that installed hardware is operational (for example, after a new SIMM is installed), but is probably unnecessary if you're sure that all problems have been rectified with the machine. So you can simply set `diag-switch?` to `false` by typing the following:

```
setenv diag-switch? false
```

Adding a second IP address without another network card

What if I want to skimp on the new network interface card, and simply add a second IP address to my existing card? I couldn't find a way to do this last time I tried (Solaris 2.4).

It's certainly possible in the later versions of Solaris, and can be very useful for hosting multiple virtual domains on a single machine, rather than relying on application-level redirections (for example, using Apache Webserver or the WU-FTP daemon). Just create a /etc/server.hme0:*n* file for each IP address, where *n* represents the virtual interface number. If you have difficulties, it can be worthwhile to get a list of the current TCP/IP settings using the `ndd` command:

```
unix% ndd -get /dev/tcp
```

In addition to retrieving selected kernel configuration parameters for network interfaces, `ndd` can also set these parameters. For further information on optimizing network interface parameters, see Adrian Cockroft's book *Sun Performance and Tuning*, published by Sun Microsystems Press. If you're running the RARP daemon, you may also need to update the ethers database located in /etc/ethers, which associates hostnames with ethernet addresses.

About our contributors

Clayton E. Crooks II is a self-employed computer consultant living in Knoxville, TN. He's married with one child. His hobbies include game development, 3-D modeling, and any athletic activity he can find time for.

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, IPG, etc. He can be reached at edd@danielyan.com.

Boris Loza holds a Ph.D. in computer science. He worked as a UNIX administrator and developer for 10 years. Currently he's working for Fidelity Investments Canada in the position of Data Security and Capacity Planner, doing IT security for UNIX, Windows NT, and Novell NetWare. He has a daughter, Anna, likes reading computer and mystery books, and likes watching movies. You can reach Boris at Boris.Loza@FMR.com.

Paul A. Watters is the project manager at Neuroflex, where he's responsible for developing natural language database systems in Java on the Solaris platform. 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 journals@zdu.com.

Editorial

Editor	Garrett Suhm
Assistant Editor	Jill Suhm
Managing Editor	Michelle Rogers
Copy Editors	Rachel Krayer Christy Flanders
Contributing Editors	Clayton E. Crooks II Edgar Danielyan Boris Loza Paul A. Watters
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) 214-2387

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 Loaicono
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

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
- Free Enterprise Javabean Servers

PERIODICALS MAIL

2096

*************3-DIGIT 480
C: 7661905 00002096 04/01

[redacted]

21
50

Please include account number from label with any correspondence.

Configuring Solaris to recognize a second network interface card

I want to add a second network interface card to my E450, which is attached to a different subnet, to act as a router. How do I configure Solaris to recognize it?

Only two changes are required, as long as the standard network initialization scripts in /etc/rc2.d (for example, S69inet), which use ifconfig, exist and are configured correctly. First, create a new device file for the new network in the /etc directory. For example, for a host called external, create the file /etc/external.hme1, if /etc/external.hme0 already exists.

In addition, you may need to add a separate netmask entry to your /etc/netmasks file, depending on the class of your network (A, B, or C), and whether the two subnets are located in different classes of the network. The hosts database (/etc/hosts) can also be updated to reflect the changes. For example, if your existing network interface has an IP address of 10.16.100.1, and the new interface has an IP address of 10.16.101.1, the host's database may include

```
10.16.100.1 external
10.16.100.1 internal
```

Boot using the 32-bit kernel

I installed the 64-bit kernel on my Ultra 2 during a recent upgrade to Solaris 7, but for purposes of performance measurement of my RDBMS, I want to boot the machine using the 32-bit kernel. How can I achieve this?

Users of high-performance applications are often concerned about quantifying performance improvements when using a 64-bit kernel compared to the old 32-bit kernel. Applications that take advantage of the greater address space provided by the 64-bit operating system should be able to do more than a limited 32-bit kernel. However, benefits in speed may not be apparent. In addition, 32-bit applications can run natively on the 64-bit operating system, so it may be unnecessary to boot the old kernel. Some users like to see for themselves, in which case the 32-bit kernel can be booted from the OpenBoot prompt by typing *boot kernel/unix*.

The Solaris Q & A column is written by Paul A. Watters. If you have a question about Solaris, send it to pwatters@mpce.mq.edu.au.

ZD Journals
Contacting Customer
Relations

If you have questions or concerns about your subscription, you can contact our Customer Relations Department by sending email to journals@zdu.com. You can also contact us by phone at 800-223-8720. Be sure to include or have on hand your customer number when you contact us. Doing so will help us to quickly and easily assist you.