

Inside Solaris™

Tips & Techniques for users of Sun Solaris

Installing GNU tools the easy way

by Don Kuenz

A great, free software site exists at www.sunfreeware.com shown in **Figure A**. At this site, you can download many popular software packages, including the GNU development tools and Perl. A couple of things make this site truly great. First, it offers a good selection of software for both SPARC and Intel platforms. Second, most software comes bundled in a compiled (binary) package, which makes installation extremely easy, provided that you know how to use Sun's package installation tools. In this article, we'll show you how to install the GNU tools available from this site.

The world's research community favors the GNU tools for software development (perhaps due to their commitment to sharing

knowledge). Also, commercial companies, such as Netscape, use GNU tools to create popular software. Given the sheer quantity of available software written with GNU tools, it makes sense to install the GNU tools even if you already own Sun's native compiler. When compiling GNU source, you'll find it much easier to use the GNU tools instead of porting GNU source to Sun's native compiler.

Both compilers can peacefully co-exist on the same host. Traditionally, you use `make` and `cc` to invoke Sun's native compiler and `gmake` and `gcc` to invoke GNU's compiler. Let's see how easy it is to install the GNU tools on your host.

Before you install the tools, you may need to create a directory named `/usr/local/bin` and include it in your `PATH` environmental variable. We'll store all of the binaries in that directory. Although you could use another directory, such as `/opt/GNU/bin`, we'll stick with `/usr/local/bin` because the GNU tools use that as the default directory. If you choose to install into a different directory, make sure you understand the `-R` option of `pkgadd`.

In an attempt to minimize bandwidth requirements, the folks behind www.sunfreeware.com provide most software in GNU zipped (gzip)

In this issue:

1 Installing GNU tools the easy way

2 System accounting

The **10** sec. TIME SAVER

7 Better back up that file—Just In Case

8 Quick Tip: Looking out for setuid programs

9 TCP_wrapper: do I need one?

11 Server-side Java

15 Solaris Q&A: What's MD5 and why should I care?

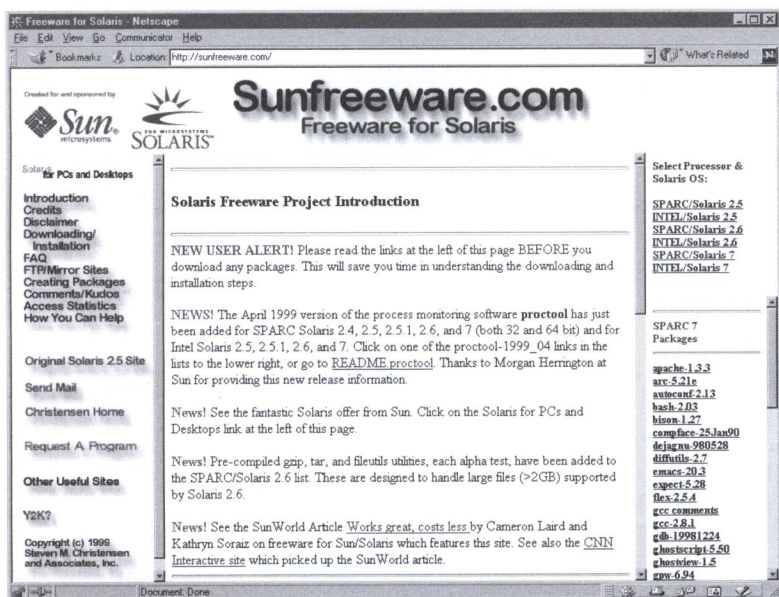


Figure A: Point your browser to the Sun Freeware site.


format. However, they do provide unzipped, binary versions of the gzip program for all Solaris 2.6 and Solaris 7 platforms. That's good news, because otherwise you would face the chicken or egg dilemma of trying to use gzip to unzip itself.

Locate the gzip version for your platform and download it to a temporary directory of your choice. Next, start a shell and make your temporary directory the current directory. Finally, use `pkgadd -d gzip*` to install gzip into `/usr/local/bin` and `pkgchk -d gzip* GNUgzip` to check your installation. `pkgchk` tells you if everything checks out.

At this point, you should download and install the GNU make (gmake) program. Download the version for your platform into

your temporary directory and use `gzip -d make*` to unzip it. Next, use `pkgadd -d make*` and `pkgchk -d make* GNUmake` to install and check it. Finally, rename `make` to `gmake` using `mv /usr/local/bin/make /usr/local/bin/gmake` to keep it separate from Sun's native make program.

You can now download and install the GNU compiler (gcc). Download it; then unzip it with `gzip -d gcc*`. Install and check it using `pkgadd -d gcc*` and `pkgchk -d gcc* GNUcc`.

Voila, you've just installed the GNU tools the easy way. Thank the good people behind www.sunfreeware.com for making things so simple. In earlier days, you would have spent the better part of a day using programs like `bison` to create `gcc` from source code. 

System accounting

by Don Kuenz

Solaris comes with several accounting programs and scripts that allow you to collect usage information about your system. You can use this information to monitor your system, to track security break-ins, and to bill users for resource usage. Although today's cheap computers may lessen the need to bill users, in an increasingly interconnected world, it pays to keep track of who uses your computer. Sometimes, accounting provides the only clue to a hacker attack.

Installing accounting costs you some disk space because it writes an audit record of certain critical events. Accounting also causes a slight decrease in performance because it records an event when every process ends. This article shows you how to install and use the accounting functionality that Sun bundles with Solaris.

Installing the packages

Sun names the two accounting packages `SUNWaccr` and `SUNWaccu`. `SUNWaccr` contains root programs, which collect usage data. `SUNWaccu` contains user programs, which report usage data. You'll probably need to install both of these packages, even if you chose to install everything during your initial Solaris installation.

You'll find both packages on your Solaris installation media. You can add them by logging on as root, mounting your Solaris CD-ROM, and invoking the `pkgadd` command. For instance, on a Solaris 7 for SPARC system, you could use the following command:

```
pkgadd -d /cdrom/cdrom0/s0/Solaris_2.7/  
Products SUNWaccr SUNWaccu
```

Among other things, the `SUNWaccr` package creates a file named `/etc/init.d/acct`. This file contains a shell script, which starts and stops the usage logging programs. Your next task is to integrate this shell script into your startup and shutdown procedures by creating a link in the `/etc/rc0.d` directory, and another link in the `/etc/rc2.d` directory.

You might recall that whenever Solaris enters a state `n`, it executes the scripts in `/etc/rcn.d`. We want to stop accounting when the system shuts down (enters state 0). You can do this by adding a link to the `/etc/rc0.d` directory using the following command:

```
ln -s /etc/init.d/acct /etc/rc0.d/K22acct
```

Likewise, you need to start accounting when the system starts multi-user mode (enters state 2). You can do this by adding a link

to the /etc/rc2.d directory with the following command:

```
ln -s /etc/init.d/acct /etc/rc2.d/S22acct
```

An astute reader might notice that we used the same script for both cases. You may ask, "How can we use the same script to start *and* stop accounting?" The answer lies with Solaris' startup and shutdown procedures. Solaris passes a stop argument to scripts that begin with the letter K. It passes a start argument to those scripts that begin with the letter S. So, the /etc/init.d/acct script just looks at its argument to see if it should start or stop the usage logging programs.

After you create those links, you need to set up a couple of batch jobs. Solaris uses a program named cron to periodically invoke batch jobs on behalf of users. The user maintains a table named crontab to tell cron when to run a job. We need to add a few jobs to the crontabs owned by root and adm.

Solaris uses a program named dodisk to collect disk usage information. You should run dodisk as root once a week. To accom-

plish this, log on as root and run crontab -e to append a line executing dodisk to the bottom of root's crontab. **Listing A** shows an example. The last line tells cron to execute dodisk at 10:00 P.M. every Thursday night.

You also need to log on as adm and append three lines to adm's crontab. **Listing B** shows the three new lines. To ensure that adm's jobs correctly execute, issue the following commands:

```
chmod 644 /var/spool/cron/crontabs/adm
chown root /var/spool/cron/crontabs/adm
chgrp sys /var/spool/cron/crontabs/adm
```

Place the following commands into adm's profile file:

```
PATH=/usr/lib/acct:/bin:/usr/bin
export PATH
```

Let's talk about the purpose of each of these three jobs. The first job, ckpacct, runs every hour to check the size of a file named /var/adm/pacct (more about this file later). Whenever pacct exceeds 500 blocks, ckpacct renames it to pacct[n] (where n is the next available

Listing A: The contents of root's crontab file

```
#ident "@(#)root 1.19 98/07/06 SMI" /* SVr4.0 1.1.3.1 */
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when daylight savings time changes.
#
# Day of Day of
#Min Hour Month Month Week Command
10 3 * * 0,4 /etc/cron.d/logchecker
10 3 * * 0 /usr/lib/newsyslog
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
0 10 * * 0,2,3,4,5,6 [ -x /usr/local/bin/daily.sh ] && /usr/local/bin/daily.sh
0 10 * * 1 [ -x /usr/local/bin/weekly.sh ] && /usr/local/bin/weekly.sh
30 22 * * 4 [ -x /usr/lib/acct/dodisk ] && /usr/lib/acct/dodisk
```

Listing B: The contents of adm's crontab file

```
#ident "@(#)adm 1.5 92/07/14 SMI" /* SVr4.0 1.2 */
#
# The adm crontab file should contain startup of performance collection if the profiling and performance feature has
# been installed.
#
# Day of Day of
#Min Hour Month Month Week Command
0 * * * * [ -x /usr/lib/acct/ckpacct ] && /usr/lib/acct/ckpacct
30 2 * * * [ -x /usr/lib/acct/runacct ] && /usr/lib/acct/runacct 2> /var/adm/acct/nite/fd2log
30 9 * * 5 [ -x /usr/lib/acct/monacct ] && /usr/lib/acct/monacct
```

sequential number) and creates a new, empty `pacct` file. The second job, `runacct`, runs early each morning and creates summary files, which the accounting report programs use as input. It creates these files under the `/var/adm/acct` directory. The third and final job, `monacct`, runs each Thursday morning and creates a report based on information found in the `/var/adm/acct/sum` directory, before cleaning up that directory.

Your final installation task consists of updating a file named `/etc/acct/holidays`. **Listing C** shows the contents of this file. To avoid nagging error messages, make sure that your file contains the correct current year. You may also

change your prime time (heavy usage) hours and add other holidays as you see fit.

Usage collection

After you install accounting, Solaris starts writing usage information to several files in the `/var/adm` directory. This article discusses three important binary files stored in that directory named `wtmp`, `pacctn`, and `acct/nite/diskacct`. **Listing D** contains file excerpts, which show the data contained in each of the three files. Let's discuss how your system creates each of these files and what it stores in them.

The first structure in **Listing D** (`utmp`) shows the format of data contained in the `wtmp` file. This file contains connection and change of state information. Solaris programs store logins, date changes, reboots, and shutdowns in this file.

The next structure (`acct`) shows the format of data contained in the `pacct` file. This file contains process information. Every time a process ends, the kernel writes one record to this file.

The third structure shown in **Listing D** (`tacct`) shows the format of the `acct/nite/diskacct` file, which holds information about disk usage. The `dodisk` job, from `root's crontab`, takes a snapshot of disk usage and updates this file every Thursday at 10:30 P.M.

Reports

You can produce several reports from the usage data collected by Solaris. We discuss a few of the reports in this article. You may peruse ac-

Listing C: The contents of the `/etc/acct/holidays` file

```
* @(#)holidays January 1, 1998
*
* Prime/Nonprime Table for UNIX Accounting System
*
* Curr   Prime Non-Prime
* Year   Start Start
*
*      1999   0800   1800
*
* only the first column (month/day) is significant.
*
* month/day Company
*      Holiday
*
* 1/1       New Years Day
* 7/4       Indep. Day
* 12/25     Christmas
```

Listing D: Include file excerpts, which show the format of three usage data files


```
/*-----*/
/* excerpt from /usr/include/utmp.h */
/*
* This data structure describes the utmp entries returned by
* the getutent(3c) family of APIs. It does not (necessarily)
* correspond to the contents of the utmp or wtmp files.
*
* Applications should only interact with this subsystem via
* the getutxent(3c) family of APIs, as the getutent(3c) family
* are obsolete.
*/
struct utmp {
    char ut_user[8];           /* User login name */
    char ut_id[4];            /* /etc/inittab id(usually line #) */
    char ut_line[12];         /* device name (console, lnxx) */
    short ut_pid;             /* short for compat. - process id */
    short ut_type;            /* type of entry */
};
```


counting documentation to find out about even more reports.

You use a program named `acctcom` to produce the first report, which is really more of a list. The `acctcom` program supports a number of options that enable you to format its output in many ways. **Listing E** shows an excerpt of output produced by an `acctcom -a` command. Keep in mind that this program reads the `pacct` file, which contains a record from every process executed by the system. As such, it can grow to become quite large. We took the liberty of editing out the middle records from the output shown in **Listing E**, which explains why the time suddenly jumps from 02:30:02 to 03:00:00. You can see that the very first record contains information about the `accton` process that turns on accounting during system startup. The `-a` option caused `acctcom` to print the last line, which displays some average statistics. The remaining options allow you to tailor its output so that you can do things like pipe the output into a filter program and look at the activities of a single user.

You use a program named `/usr/lib/acct/prdaily` to print the daily reports shown in **Listing F** on page 6. Again, we deleted some lines to conserve space. As you can see, these reports provide an excellent picture of user activity on your system.

Summary

Solaris's accounting functionality provides useful information for monitoring your system, detecting break-ins, and, to a lesser extent, billing users. This article shows you how to install and use two accounting packages, which Sun bundles with Solaris. 

Listing D: *continued*

```

struct exit_status ut_exit; /* The exit status of a process */
                          /* marked as DEAD_PROCESS. */
time_t ut_time;          /* time entry was made */
};

/*****
/* excerpt from /usr/include/sys/acct.h */
*****/

struct acct {
char    ac_flag;        /* Accounting flag */
char    ac_stat;        /* Exit status */
uid32_t ac_uid;         /* Accounting user ID */
gid32_t ac_gid;         /* Accounting group ID */
dev32_t ac_tty;         /* control typewriter */
time32_t ac_btime;     /* Beginning time */
comp_t  ac_utime;       /* acctng user time in clock ticks */
comp_t  ac_stime;       /* acctng system time in clock ticks */
comp_t  ac_etime;       /* acctng elapsed time in clock ticks */
comp_t  ac_mem;         /* memory usage */
comp_t  ac_io;          /* chars transferred */
comp_t  ac_rw;          /* blocks read or written */
char    ac_comm[8];     /* command name */
};

/*****
/*
* total accounting (for acct period), also for day
*/
*****/

struct tacct {
uid_t   ta_uid;         /* userid */
char    ta_name[8];     /* login name */
float   ta_cpu[2];      /* cum. cpu time, p/np (mins) */
float   ta_kcore[2];    /* cum kcore-minutes, p/np */
float   ta_con[2];      /* cum. connect time, p/np, mins */
float   ta_du;          /* cum disk usage */
long    ta_pc;          /* count of processes */
unsigned short ta_sc;   /* count of login sessions */
unsigned short ta_dc;   /* count of disk samples */
unsigned short ta_fee;  /* fee for special services */
};

```

Listing E: *Output from the acctcom command*

COMMAND	START	END	REAL	CPU	MEAN
NAME	TIME	TIME	(SECS)	(SECS)	SIZE(K)
#accton	02:30:03	02:30:03	0.06	0.06	258.67
turnacct	02:30:02	02:30:02	0.63	0.05	241.60
ckpacct	03:00:00	03:00:01	1.10	0.10	288.00
#sh	03:00:00	03:00:01	1.49	0.18	564.44
acctcom	03:00:15	03:00:15	0.44	0.44	826.91
ls	03:00:43	03:00:43	0.10	0.10	309.20
ls	03:00:55	03:00:55	0.11	0.10	389.20
cmds=144 Real=0.58 CPU=0.09 USER=0.03 SYS=0.07 CHAR=2578.76 BLK=0.34 USR/TOT=0.28 HOG=0.16					

Listing F: Output from the prdaily program

Apr 25 03:19 1999 DAILY REPORT FOR hyperion Page 1

from Tue Apr 20 04:26:45 1999
 to Sun Apr 25 02:30:03 1999
 1 run-level 6
 1 system boot
 1 run-level 3
 1 runacct
 1 acctcon

TOTAL DURATION IS 7083 MINUTES

LINE	MINUTES	PERCENT	# SESS	# ON	# OFF
/dev/pts/1	0	0	0	0	1
/dev/pts/2	0	0	0	0	0
console	576	8	8	8	17
pts/1	101	1	3	3	4
pts/3	0	0	0	0	2
pts/2	132	2	3	3	3
TOTALS	809	--	14	14	27

Apr 25 03:19 1999 DAILY USAGE REPORT FOR hyperion Page 1

LOGIN	CPU (MINS)	KCORE-MINS	CONNECT (MINS)	DISK	# OF	# OF	# OF	FEE
UID NAME	PRIME	NPRIME	PRIME	NPRIME	PRIME	PRIME	SESS	SAMPLES
0 TOTAL	0 1 0 538	0	809	0	549	13	0	
0 root	0 0 0 67	0	47	0	51	6	0	
4 adm	0 1 0 303	0	132	0	396	2	0	
100 don	0 0 0	168	0	630	0	102	0	
101 jean	0 0 0	0	0	0	0	1	0	

Apr 25 02:30 1999 DAILY COMMAND SUMMARY Page 1

COMMAND NAME	NUMBER CMDS	TOTAL KCOREMIN	TOTAL COMMAND SUMMARY						
			TOTAL CPU-MIN	TOTAL REAL-MIN	MEAN SIZE-K	MEAN CPU-MIN	HOG FACTOR	CHARS TRNSFD	BLOCKS READ
TOTALS	549	535.86	1.09	486.56	491.39	0.00	0.00	8460250	248
sendmail	40	72.19	0.08	0.18	941.66	0.00	0.43	162234	37
sh	27	65.43	0.11	239.14	616.33	0.00	0.00	144113	0
apropos	4	53.64	0.08	0.08	690.68	0.02	0.97	2804672	1
ls	54	39.73	0.10	0.11	401.29	0.00	0.92	36557	1

Apr 25 02:30 1999 MONTHLY TOTAL COMMAND SUMMARY Page 1

COMMAND NAME	NUMBER CMDS	TOTAL KCOREMIN	TOTAL COMMAND SUMMARY						
			TOTAL CPU-MIN	TOTAL REAL-MIN	MEAN SIZE-K	MEAN CPU-MIN	HOG FACTOR	CHARS TRNSFD	BLOCKS READ
TOTALS	549	535.86	1.09	486.56	491.39	0.00	0.00	8460250	248
sendmail	40	72.19	0.08	0.18	941.66	0.00	0.43	162234	37
sh	27	65.43	0.11	239.14	616.33	0.00	0.00	144113	0
apropos	4	53.64	0.08	0.08	690.68	0.02	0.97	2804672	1
ls	54	39.73	0.10	0.11	401.29	0.00	0.92	36557	1

Listing F: Continued

Apr 25 02:30 1999 LAST LOGIN Page 1

```
00-00-00 bin          00-00-00 nobody      99-04-25 adm
00-00-00 daemon        00-00-00 nobody4     99-04-25 don
00-00-00 listen         00-00-00 nuucp       99-04-25 jean
00-00-00 lp             00-00-00 sys         99-04-25 root
00-00-00 noaccess       00-00-00 uucp
```

Better back up that file— Just In Case

The **10** sec.
TIME SAVER

by Arthur Haigh

Over the years I've learned that before I modify a file, it's always a good idea to make a backup copy first. I like the convention of copying the file and renaming, taking the original filename, adding a dot, and then appending the date. For example, say I want a backup copy of the password file. Usually I'd issue commands like the following:

```
# date
Fri Feb 12 14:35:48 EST 1999

# cp -p /etc/passwd /etd/passwd.021299
```

I've written a quick script that automates this procedure. The script, which I've named `jic` (for Just In Case), is shown in [Listing A](#) on page 8. The usage is simply

```
# jic filename
```

where *filename* is the name of the file you want to back up.

Examining the script

The script is simple. First, using the case statement (line 10, [Listing A](#)), the script checks the syntax for the proper usage. The usage is the script name (`jic`) and then the filename of the file to be backed up. The case statement ignores the script name and counts only the entries following `jic`. If the number of entries after `jic` is 1 (line 11), then the script assumes that the parameter is a filename and checks, essentially, for the existence of this file (line 12). If the file exists, `jic` creates the backup filename (lines 16 and 17) and then checks for the existence of the new filename (line 19). The

copy command is executed (line 30) if the backup copy filename doesn't exist. If either the original filename doesn't exist (lines 12, 13 and 14) or the copy filename exists (lines 19, 20, and 21), the shell aborts. If the number of parameters given on the command line is anything other than 1 (line 25), then the shell issues the usage error message and exits (lines 26 and 27). This is what you'll see:

```
# ls -l /etc/passwd*
-r--r--r-- 1 root sys 462
Feb 10 17:17 /etc/passwd

# jic /etc/passwd
# ls -l /etc/passwd*
-r--r--r-- 1 root sys 462
Feb 10 17:17 /etc/passwd
-r--r--r-- 1 root sys 462
Feb 12 15:33 /etc/passwd.021299
```

Taking it one step further

As I used this tool, I found that I nearly always made a backup because I was about to edit the file. I prefer using the `vi` editor, so I built that next step, editing the original file, into the program. I've included the line calling `vi` in the `jic` script (line 31). You can see that it's commented out. If you'd like to add the `vi` option, just remove the pound sign (`#`) from the beginning of the line.

One more option

You may have noticed that the naming convention for the backup file has a shortcoming: the program will abort if you try to back up a file more than one time per day. You can

Listing A: JIC is a short script to make a quick backup copy of a file.

```
1 #!/bin/sh
2 #-----
3 #
4 # jic: Makes a copy of a file & appends the date.
5 # with an option to open the original file in the vi editor
6 #
7 # usage: jic filename
8 #
9 #-----
10 case "$#" in
11     1) # 1 parameter specified
12         if [ ! -f "$1" ] ; then          # filename doesn't exist
13             echo "error: No such file $1"
14             exit
15         else
16             FILE="$1"                    # filename is OK
17             FILE_BUP="$FILE.`date +%m%d%y`"
18             FILE_BUP="$FILE.`date +%m%d%y%H%M`"
19             if [ -f $FILE_BUP ] ; then    # new name unique ?
20                 echo "error: $FILE_BUP exists, abort"
21                 exit
22             fi
23         fi
24     ;;
25     *) # Illegal number of parameters
26         echo "\nerror: usage: $0 filename\n"
27         exit
28     ;;
29 seac
30 /usr/bin/cp $FILE $FILE_BUP            # make the backup copy
31 #/usr/bin/vi $FILE                      # option to open vi
exit
```


modify the shell script to simply overwrite any existing backup copy that already exists by commenting out lines 19 through 22. Another option is to leave lines 19 through 22 alone, comment out line 17, and uncomment line 18. Line 18 creates the backup copy filename by appending the date and the time of day to the original filename. In this case, the backup passwd file would be called

```
/etc/passwd.0212991533
```

This makes the filenames a bit long, but you're assured that you won't overwrite your files (as long as you don't try to create multiple backups per minute).

To allow for simple execution, you can place the script in a common directory like `/usr/local/bin`. Be sure that `/usr/local/bin` is in your search path. If it's in your search path, then you'll need to refresh the hash table before your shell will recognize it. If you're using the C shell, then issue the `rehash` command. For the Bourne shell, use `hash`. See the `rehash` man page for more information.

Conclusion

In my experience, any systems administration tool must be simple and convenient. The Just In Case shell script satisfies these requirements and provides a safeguard against overwriting important files. 

Looking out for setuid programs

by Al Alexander

Perhaps you've heard of a setuid (SUID) program before. If you haven't, a *setuid* program is one that lets a user run the program with the permission of the program owner. For example, if you create a setuid program, and make the owner of the program root, any person that runs the program will run the program with root user permissions. Setuid programs are needed for many applications, but, as you can imagine, they also create security problems.

In a recent break-in of an Internet provider, the hackers broke into a system and stayed logged on to the system as the root user just long enough to create a SUID program. When they logged on to the system again later as another user, they used their SUID program to achieve their mischief.

As an administrator, you want to keep a lookout for setuid programs. An easy way to do this is by

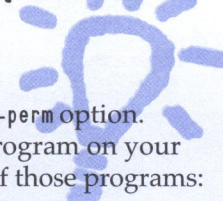
using the `find` command with the `-perm` option. Here's how to find every setuid program on your system, and generate an `ls` listing of those programs:

```
find / -perm -4000 -exec ls -ld {} \; >
➔ /tmp/suid.files
```

This simple one-liner searches your file system for all setuid programs and prints the listing to the file named `/tmp/suid.files`. There are a fair number of setuid files on a basic Solaris system, so the key is really to look at how this listing changes from day to day, week to week, or month to month (depending on your security concerns).

If one day you suddenly find a new setuid program on your system, and it's owned by root—watch out. You'll want to disable that program in a hurry.

QUICK TIP



TCP_wrapper: do I need one?

by Boris Loza

The brave new world of the Internet has brought many dangers as well as opportunities. Security is now a bigger concern than ever, with hackers creating viruses daily. It makes sense to protect your computing resources every way you can, and TCP_wrapper is one of these tools that can help you. This article will explore what TCP_wrapper can do for you.

Understanding TCP_wrapper

The *TCP_wrapper* is a tiny utility program written by Wietse Venema that allows you to control access to Internet services such as ftp, telnet, rlogin, tftp, finger, rsh, and systat. When the service is requested, TCP_wrapper decides first whether to allow or deny access for this request.

TCP_wrapper is based on the approach that TCP/IP networks use a client/server model. For instance, when you decide to log on to a remote system using the rlogin command, it sends a request to the rlogind process on the server. This request goes through the inetd daemon, which, in turn, checks the /etc/services file for the name of a particular service and the port number. Finally, the /etc/inet/inetd.conf is searched for the name of the program or daemons providing the service.

If you type *rlogin one.host.com*, the rlogin client on your machine sends a connection request to the server one.host.com with the information about the user and terminal. Because rlogin uses the TCP port 513, inetd looks up this port in /etc/services and finds the service name login. Looking up login in /etc/inet/inetd.conf inetd finds that it needs to run the daemon in.rlogind. After in.rlogind responds to the client, the rlogin session is started.

If you want to control network access to your machine, you can use TCP_wrapper. Instead of directly running the in.rlogind daemon, inetd runs the TCP_wrapper program to perform the source and the destination address checks to decide whether this particular host and user are allowed to communicate with your machine. Finally, it writes a note in the system log about this connection.

Finding and installing the program

A copy of TCP_wrapper can be obtained by ftp at ftp://ftp.cert.org/pub/tools/tcp_wrappers/tcp_wrappers_7.6.tar.gz.

There are two ways to install the TCP_wrapper program: easy and advanced. For the easy installation, move your network daemons to some other directory and fill the resulting holes with copies of the TCP_wrapper programs. This approach doesn't require any changes in the /etc/inet/inetd.conf file. For the advanced installation, leave the network daemons alone and modify the inetd.conf file. In this article, we'll use the advanced method.

First, unpack *tcp_wrappers_7.6.tar.gz* into the desired directory. Go to the *tcp_wrappers_7.6* directory and run the *make* command with the following options:

```
#make REAL_DAEMON_DIR=/usr/sbin \  
sunos5
```

If you have a GNU gcc compiler, you may want to specify:

```
#make CC=gcc \  
REAL_DAEMON_DIR=/usr/sbin \  
sunos5
```

This will tell TCP_wrapper that all Internet daemons (in.telnetd, in.ftpd, in.fingerd, in.rlogind, etc.) are located in the /usr/sbin directory, the C compiler is GNU gcc, and the operating system is Solaris. Be sure to check the default compile options in the Makefile first. For instance, the PARANOID compile option tells TCP_wrapper to always attempt to look up and double check the client host name. It will always refuse service in case of a host name/address discrepancy. For more options, read the Makefile.

Configuring TCP_wrapper

Now you need to modify the inetd.conf file to tell the inetd daemon that TCP_wrapper is in use. Find the service you would like to control and advise it to use tcpd. In the following example, we're going to make ftp, telnet, and login services use TCP_wrapper. We'll modify the following lines in the /etc/inetd.conf file:

```
ftp stream tcp      nowait root \  
➔ /usr/sbin/in.ftpd  in.ftpd  
  
telnet stream tcp   nowait root \  
➔ /usr/sbin/in.telnetd in.telnetd  
  
login stream tcp    nowait root \  
➔ /usr/sbin/in.telnetd in.rlogind
```


Put in the location of the `tcpd` daemon, instead of the location of the service daemon:

```
ftp stream tcp      nowait root \
➤ /usr/etc/tcpd    in.ftpd

telnet stream tcp   nowait root \
➤ /usr/etc/tcpd    in.telnetd

telnet stream tcp   nowait root \
➤ /usr/etc/tcpd    in.rlogind
```

Don't forget to send `kill -HUP` to the `inetd` process to make the changes effective.

Now you have to deal with two `TCP_` wrapper configuration files: `/etc/hosts.allow` and `/etc/hosts.deny`. You need the `/etc/hosts.allow` file to allow connections to your computer and the `/etc/hosts.deny` file to deny access. If you'd like to allow access to all Internet services on your machine to everyone, leave the `/etc/hosts.allow` and the `/etc/hosts.deny` files empty. This configuration is known as *Mostly Open*.

Assume you want to allow `one.trust.host` access through `telnet` only. In this scenario, you have to modify the `/etc/hosts.allow` file as follows:

```
ALL      : LOCAL
in.telnetd : one.trust.host
```

and the `/etc/hosts.deny` file:

```
ALL      : ALL
```

The first line in the `/etc/hosts.allow` tells the `tcpd` daemon that an access is allowed for all machines in our local domain. Second, only `telnet` service is allowed from the `one.trust.host`. The `/etc/hosts.deny` file tells the `tcpd` daemon to close all services for any other machine. In `hosts.allow`, we list only those specific services we want others to use. The pound sign (`#`) could be used as a comment.

One more example: suppose you need to deny access to anyone not in your `/etc/hosts` file to `telnet` and `rlogin` services. The `hosts.deny` file would be:

```
in.telnetd in.rlogind : UNKNOWN
```

We use the `TCP_wrapper`'s wildcard `UNKNOWN` that matches any IP address that doesn't have a corresponding hostname. Other wildcards that can replace specific host names are `KNOWN` and `PARANOID`. `KNOWN` matches

any IP address that has a corresponding host-name, and `PARANOID` matches any host whose name doesn't match its Internet address. For the complete list of wildcards, see man pages for `hosts_access(5)`.

After creating rules, you can check the configuration files with the `tcpdchk` and `tcpdmatch` commands. The first one checks the configuration files for any problems. It will tell if you've used wildcards incorrectly, if there are non-corresponded host names in the access file, and much more. For example:

```
#tcpdchk -v
Using network configuration file: /etc/
➤ inet/inetd.conf
>>> Rule /etc/hosts.allow line 1:
daemons: ALL
clients: LOCA
warning: /etc/hosts.allow, line 1: LOCA:
➤ host not found
access: granted
>>> Rule /etc/hosts.deny line 1:
daemons: in.telnetd in.rlogind
clients: UNKNOWN
access: denied
```

In this example the wildcard `LOCAL` in the `/etc/hosts.allow` file is misspelled. The `tcpdchk` utility expects it to be a host name, but can't find this name in the `/etc/hosts` file.

The `tcpdmatch` utility tests the configuration files against a virtual request for an Internet connection. You have to provide the name of the daemon and a host name, and it tells you whether that connection would be allowed or denied:

```
#tcpdmatch in.rlogind one.untrust.host

client: address one.untrusted.host
server: process in.rlogind
matched: /etc/hosts.deny line 1
access: denied
```

For more details, see the man pages for `TCP_wrapper`.

What else can I do with it?

Another useful feature of `TCP_wrapper` is that it logs every request for a connection, whether it's granted access or not. By default, the wrapper logs go to the same place as the transaction logs of the `sendmail` daemon. So, if your mail messages are written into the `/var/log/`

syslog file, you can grep this file for telnetd or rlogind messages to check who has been trying to connect.

In addition, you can run shell commands using "booby traps" when certain services are requested. Assume that you want to know exactly who did a telnet into your machine. In this scenario, add the following line into the `/etc/hosts.deny` file:

```
in.telnetd: ALL: (/usr/sbin/safe_finger -l
➔ \ @%h | /usr/ucb/mail -s %d-%h root) &
```

The `safe_finger` command comes with `TCP_wrapper`, and as the `TCP_wrapper` README says, it "gives better protection against nasty stuff that remote hosts may do in response to your finger probes." The strings `%h` and `%d` are called expansions, and `tcpd` replaces them with the corresponding text for the host name and daemon file and sends to root. There are other expansions that can be used in booby traps (see man pages for `hosts_access(5)`).

Instead of specifying a shell command that should be executed, `TCP_wrapper` allows you to specify a set of options. To use this feature, you have to compile the `TCP_wrapper` with the option `DPROCESS_OPTIONS`. After doing this, you can have in the configuration files a new format of the access control rules:

```
daemon : host : option
➔ : option ...
```

Options include `allow`, `deny`, and many others.


Suppose you wish to deny all connections to your server, except from the host `my.host.com`. You just need to put two lines into the `hosts.allow` file:

```
telnetd,rlogind : my.host.com : allow
all : all : deny
```

In this case, we can get rid of the `hosts.deny` file completely. You can find more information about these features in the `hosts_options(5)` man page.

Should I use TCP_wrapper?

`TCP_wrapper` is a very powerful program that could arm your system with a good access control solution. If you use the secure shell (`ssh`) it comes with Wietse Venema's `TCP_wrapper` package. If `ssh` is compiled with the `TCP_wrapper`, you can control the basic `ssh` connections, X11 forwarding, and `TCP` port forwarding.

As useful as `TCP_wrapper` sounds, keep in mind that it has some limitations. First of all, `TCP_wrapper` can't protect services started at boot time and run until system shutdown, like `sendmail` and `httpd`. Additionally, it's vulnerable to IP spoofing because it uses the IP address for authentication. (The solution for this is to keep your `/etc/hosts` file up to date and don't rely on outside DNS). But in spite of these limitations, we have found the `TCP_wrapper` to be extremely useful for securing our networked environment. 

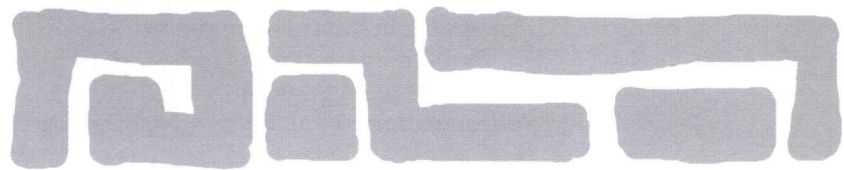
Server-side Java

by Paul A. Watters

Java is a new programming language that's often used to create pleasant graphical user interfaces (GUIs) that are platform-independent, and which can interact with the user in more complex and sophisticated ways than static HTML. However, these applets are only one side of the whole Java story; in this article, we'll look at the server side of Java. Java applications that execute on the server are called servlets, and have their own standard API specification, which has now

been widely implemented in Web server extension products known as servlet runners (for example, Live Software's `Jrun`). In this article, we'll determine the extent to which servlets are useful in developing Solaris-based enterprise applications that are Web-enabled.

Increasingly, applications in the enterprise are being implemented using Web interfaces. This is partly a response to the persistent heterogeneity of computing platforms within organizations that span cities, states, and even



nations. Accepting platform diversity doesn't mean losing control of standards; the response from Sun Microsystems has been to pioneer a platform-independent programming language, where applications run on top of a logical virtual machine (JVM) that presents a consistent API for developers.

Most major hardware platforms and operating systems now have virtual machines implemented, including Solaris (obviously). In fact, the Solaris JVM produced by Sun has been highly optimized in its production release series. JVMs have also been integrated into popular Web browsers, so that Java programs can be downloaded from a server and executed within these browsers. (HTML now has an `<applet>` tag that facilitates this process.) Applets have increased the complexity of Web-based user interfaces from simple arrays of buttons and forms to dynamic interaction with the user in a way that's similar to a normal desktop application.

Although Java has been successful in improving the client side of Web-based computing, it has been slower to make an impact on the server side. (This is as much a result of the excitement surrounding applets as any deficit in the servlet API.) However, many people believe that the server side is where Java has its greatest potential.

The notion of having platform-independent enterprise applications that run through a standard Web interface promises to change the way that users, developers, and software interact. The "write once, run anywhere" philosophy means that servers with totally different operating systems and hardware can be replaced with newer systems without concern for application stability and porting.

Commonly used Java classes can be bundled as beans, which can provide rapid implementation for our client's business logic. Full access to the Java API and database servers is also provided for Java servlets, using the Java Database Classes (JDBC) supplied by Oracle and other major vendors. These features ensure that today's Java server-side programs won't become tomorrow's legacy applications.

Java vs. CGI

How does server-side Java compare to Web-based client-server techniques, such as the combination of a Common Gateway Interface (CGI) and a non-object-oriented language such as C? Although a compiled language like C is faster on a byte-per-byte basis than an inter-

preted language like Java, performance increases for Java can be gained by the combination of optimizing just-in-time compilers for specific platforms, and by reducing the process and memory overhead associated with CGI.

For example, if we had a search application written in Perl that was accessed by 1,000 Web users per hour, that is, an extra 1,000 invocations of Perl that the server has to deal with. Of course, if we're running on an E10000, that's probably negligible. For the rest of us, invoking a Java servlet that occupies only a single process after being loaded into memory that *persists* across sessions is therefore very memory and process efficient. Servlets are therefore more appropriate for applications that are constantly being executed by multiple users, by taking advantage of Java's multi-threading and synchronization capabilities.

On the flip side, CGI programs are often better suited to single-user, infrequently used, and numerically intensive applications that might only be invoked once per hour. In addition, CGI programs written in C are logically isolated from each other in the server's memory space as separate processes. If Java servlets are executed using a single instance of a service manager (for example, Live Software's Jrun), an unhandled exception arising from malformed or unexpected input could potentially impact all servlets running through the manager. This can be especially bad if the Java Virtual Machine (JVM) itself crashes.

How do I run servlets?

One of the easiest ways to evaluate servlet technology is to download the Java Servlet Development Kit (JSDK), which is freely available from Sun's Java Web site:

www.java.sun.com/products/servlet/

The classes included with this reference implementation (like `javax.servlet.*`) can be used with the supplied stand-alone server (known as `servletrunner`), or can be called using a servlet-enabled Web server. Actually, most Web servers don't yet have builtin support for Java (excluding Sun's Web server). Typically, a third-party servlet engine must be installed. A popular combination is the free Apache Web server (www.apache.org) and Live Software's Jrun service manager, which has a free, mostly functional trial download (www.livesoftware.com). In addition to performance and reliability benefits, Jrun also supports Java Server Pages,

which have the ability to include servlet code in-line with HTML pages (a Java version of server-side includes, with many enhancements).

How do we configure Apache and Jrun to work with Solaris? There are three basic steps: compiling the Jrun connector proxy source within the Apache source tree, and activating the Jrun module (`mod_jrun`); configuring Apache to recognize servlets; and configuring individual servlets within the Jrun service manager. Let's look at what's involved in each step.

Compiling a connector

The Jrun service manager will operate with several different Web servers, including Apache. However, a different connector proxy interface is required for each type of Web server. The source for these connectors is located under the `JRUN_HOME` source tree in the `connectors` sub-directory. The connector source is copied across to a `src/modules/jrun` directory under the Apache 1.3.x source tree, and Apache is re-compiled with a command like:

```
./configure -prefix=/opt/apache \  
-activate-module=src/modules/  
↳jrun/libjrun.a
```

The `activate-module` directive should lead to this message being displayed during the Apache build:

```
+ activated jrun module (modules/  
↳jrun/libjrun.a)
```

Configuring Apache

After the Jrun-enabled version of Apache has been installed, the `httpd.conf` file can be edited to include directives for handling servlets using `mod_jrun`:

```
# JRun Settings  
<IfModule mod_jrun.c>  
JRunConfig Verbose false  
JRunConfig ProxyHost 127.0.0.1  
JRunConfig ProxyPort 8081  
JRunConfig Timeout 180  
JRunConfig Mappings "/opt/jrun-2.2.1/2.2/  
↳jrun-default/services/jse/properties/  
↳rules.properties"  
</IfModule>
```

If these settings have been copied from a version of Jrun prior to 2.2.x, it's wise to delete

the following definitions since they can cause problems on Solaris:

```
JRunConfig InitPoolSize 5  
JRunConfig MaxPoolSize 100
```

Configuring Jrun

The `rules.properties` file in the Jrun tree contains directives for handling various URLs. An example `rules.properties` file would look like:

```
*.jrun=invoker  
/servlet/=invoker
```

This means that URLs for files with the extension `.jrun` or that reside under the directory `servlet` will execute a servlet. For example, the servlet `SearchFoo` on the server `www.bar.com` could be executed by the URL: `www.bar.com/servlet/SearchFoo`.

The properties that govern the execution of servlets can be found in (unsurprisingly) `properties` files. After servlets have been loaded into the appropriate servlets directory defined in `servlets.properties`, the Jrun service manager can be started with a command like:

```
java com.livesoftware.jrun.service.  
↳ServiceManager
```

Performance tuning

In the future, when servlet runners and Web servers are fully integrated, performance tuning might be a little less the black art that it currently is. However, the tuning of Apache and Jrun isn't too difficult (and there's an excellent mailing list archive on the Live Software site with over 500 Solaris-related Jrun articles). The most important rule with Jrun is to think big—in other words, plan for the peak load periods that your servlets will have to cope with, keeping in mind that the number of processes will be reduced greatly compared to CGI as previously outlined.

In the `session.properties` file, for example, we can set the `session.invalidationinterval` and `session.invalidationtime` to be quite large (for example, 180,000 for 3 minutes). A corresponding value can be set in Apache's `httpd.conf` for the `Timeout` parameter. These values will vary depending on hardware capacity and available network bandwidth. More importantly, there are several parameters in both `httpd.conf` and the Jrun `session.properties` file that must be consistent with

each other; otherwise, concurrency overflow will arise. In this case, the values for Jrun must *exceed* those given for Apache. For example, the MaxClients 150 parameter in httpd.conf must be fewer than the endpoint.main.max.threads and endpoint.main.active.threads defined in session.properties, otherwise a dreaded error message will be displayed to users:

```
Too many concurrent users. Please try
↳again later.
```

Error logs for Jrun are also a very valuable way to isolate errors and improve performance. For example, the concurrency error can arise if the Jrun license key isn't entered correctly. So, if our Jrun log file had the entry:

Listing A: A very simple "Hello World" servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletHelloWorld extends
↳HttpServlet
{
    public void doGet (
        HttpServletRequest rq,
        HttpServletResponse rs
        ) throws ServletException,
↳IOException
    {
        PrintWriter out;
        rs.setContentType("text/html");
        out = rs.getWriter();
        out.println("<B>Hello, World!</B>");
        out.close();
    }
}
```

```
[Mon Apr 19 01:20:39 EDT 1999] Found
↳valid JSM.5 license, enabling unlimited
↳concurrency.
```


we could rest easy. However, if we received the message

```
[Tue Mar 30 01:05:44 EST 1999] Did not
↳find a valid license, disabling unlimited
↳concurrency.
```

it might explain any concurrency issues that had arisen during servlet execution.

A sample servlet

Now that we've examined how to implement a scalable servlet and Web server solution, let's take a quick peek inside a servlet. After writing libraries of C functions to parse a GET stream, it's refreshing to be able to use a single Java method to achieve the same result. A simple variation on a theme for, "Hello World," Servlet-style is shown in **Listing A**.

Although this servlet doesn't meet some of the current standards for HTML (like HEAD and BODY elements), it prints a simple greeting by extending HttpServlet and overloading the doGet() method (only HttpServletResponse is used in this servlet, however). After setting the content type to be HTML, PrintWriter is activated to produce some simple HTML output that's produced upon every request to the servlet passed by the Web server. Servlets (and Java programs in general) are much more complex than the simple example presented here, but the same principles of inheritance and overloading can result in complex applications being constructed from relatively few lines of code. 

Further reading

- The best place to read more about servlet technologies, and Java in general, is Sun's Java pages at www.java.sun.com.
- More information about the Apache Web server can be found at www.apache.org.
- Details concerning Live Software's Jrun servlet runner can be retrieved from www.livesoftware.com.

Customer Relations

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

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

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

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

Editorial

Editor.....Garrett Suhm
Assistant EditorJill Suhm
Copy Editors.....Rachel Krayer
Christy Flanders
Taryn Chase
Contributing EditorsAlvin J. Alexander
Arthur Haigh
Don Kuenz
Boris Loza
Lance Spitzner
Paul A. Watters
Print DesignersRachel J. King
Lance Teitworth

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

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

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

Editorial Department fax(716) 214-2387

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

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

Postmaster

Periodicals postage paid in Louisville, KY.

Postmaster: Send address changes to

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

Copyright

Copyright © 1999, ZD Inc. ZD Journals and the ZD Journals logo are trademarks of ZD Inc. *Inside Solaris* is an independently produced publication of ZD Journals. All rights reserved. Reproduction in whole or in part in any form or medium without express written permission of ZD Inc. is prohibited. ZD Journals reserves the right, with respect to submissions, to revise, republish, and authorize its readers to use the tips submitted for personal and commercial use.

Inside Solaris is a trademark of ZD Inc. Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, SunOS, SunInstall, OpenBoot, OpenWindows, DeskSet, OMC, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. Other brand and product names are trademarks or registered trademarks of their respective companies.

Printed in the USA.

Price

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

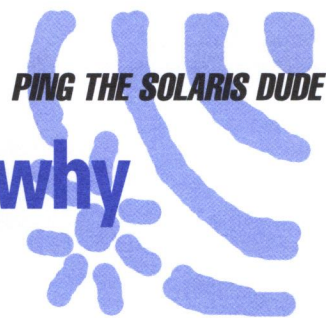
Our Canadian GST# is: R140496720. CPM# is: 1446703.

Back Issues

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

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 zdjcr@zd.com.



PING THE SOLARIS DUDE

What is MD5 and why should I care?

by Lance Spitzner

When you send data over a network, there are three issues that concern most organizations: security, authenticity, and integrity. The security of your data ensures that no one can read your data. This is important for the military, where secrets have to be kept from enemy hands. Authenticity guarantees the originator of the data; you know for certain who sent the data. This is important in the legal world, with issues such as digital signatures. Integrity guarantees that the data hasn't been altered in transit—that the data you received is the data that was sent. This is important for many industries, such as the financial world. MD5 is such a tool—it guarantees the integrity of your data.

MD5 can help you in a variety of ways. When you download files from the Internet, you can use MD5 to guarantee that you downloaded the

About our contributors

Alvin J. Alexander first began his career as an aerospace engineer. He's now the president of Mission Data Corporation, an employee-owned computer consulting firm in Louisville, Kentucky. You can reach him online at aja@missiondata.com.

Arthur Haigh is the senior systems administrator in the Division of Nuclear Medicine, School of Medicine, University of Pennsylvania. He can be reached at art@rad.upenn.edu.

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

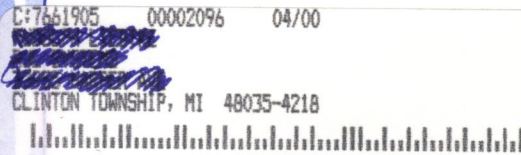
Boris Loza holds a Ph.D. in computer science from Russia. He worked as a UNIX administrator and developer for 10 years. Currently, he's working for Fidelity Investments Canada in the position of Data Security and Capacity Planner, doing IT security for UNIX, Windows NT, and Novell. He can be reached at Boris.Loza@FMR.com.

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

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

PERIODICALS MAIL

Sun Technical Support
(800) 786-7638



Please include account number from label with any correspondence.

correct file. This protects you from Trojans or corrupted files. If you use tools such as Tripwire to protect the integrity of your file system, you're most likely using MD5. You're probably using MD5 if you're using a public/private key infrastructure.

Developed in 1994, MD5 is a one-way hash algorithm that takes any length of data and produces a unique 128-bit fingerprint or message digest. No two files produce the same fingerprint. Nor is the fingerprint reversible; rather, it's computationally infeasible to determine a file based on its fingerprint. This means someone can't figure out your data based on its MD5 fingerprint. Here is an example of a MD5 output for the binary `/usr/bin/ls`:


```
homer $md5 /usr/bin/ls
MD5 (/usr/bin/ls) =
➔ 1eabd3dbc0746c8a4b5467f99a4f8823
```

The actual fingerprint is
1eabd3dbc0746c8a4b5467f99a4f8823.

Basically, what MD5 did was apply a mathematical algorithm to the `ls` binary to produce the fingerprint. (To learn the gory mathematical details about the algorithm, check out RFC 1321 at www.cis.ohio-state.edu/rfc/rfc1321.txt.) Every time you do an MD5 hash of the binary `/usr/bin/ls`, you should get the exact same fin-

gerprint. If you get a different fingerprint, then the binary has been altered, possibly because a system patch is present, or the binary has been infected with a trojan virus.

When you download a new file or patch, one of the first things you can do is an MD5 hash of the file. Compare the fingerprint to a known good fingerprint (usually posted on a remote site). If the fingerprints match, you can be assured of the file's integrity. This is how the tool Tripwire works. It builds a database of fingerprints for all your binaries, then later on compares the binaries to that database. However, tripwire uses a variety of hash algorithms in addition to MD5, such as snefru.

Since MD5 doesn't encrypt data, it isn't restricted by any exportation rules. You can freely use and distribute this tool anywhere in the world. To learn the history of MD5, check out www.rsa.com/rsalabs/faq/html/3-6-6.html. You can download MD5 at www.leo.org/pub/comp/general/security/md5/index.html. 

If you have any Solaris questions you would like answered, shoot your questions off to lance@spitzner.net, and Ping the Solaris Dude!

Coming up...

- DNS utilities
- Access control
- Indexing your backups
- Security and Solaris 7