Practical Unix security and the superuser

By Simson Garfinkel
and Gene Spafford

Editor's Note: In their recent book, "Practical Unix Security" (1991, O'Reilly & Associates Inc., Sebastopol, Calif.), Simson Garfinkel and Gene Spafford argue that a properly run Unix system can be just as secure as any other readily available system and that Unix's reputation as unsecure is a bad rap. The entire book is recommended reading for system administrators responsible for Unix systems. The following excerpt explains the concepts of users, groups and what the authors call Unix's main security flaw, the "superuser."

Although every Unix user has a username up to eight characters long, inside the computer Unix represents each user by a single number: the UID (user identifier). Usually, the Unix system administrator gives every user on the computer a different UID. Unix also uses special UIDs for system "users" such as the following:

root, the superuser, which performs accounting and low-level system functions.
uucp, which manages the UUCP system.
daemon, which handles some aspects of the network. This user also handles other utility systems, such as the print spoolers, on some versions of Unix.

UIDs are signed 16-bit numbers, which means they can range from −32768 to 32767. UIDs between 0 and 9 are typically used for system functions; UIDs for humans usually begin at 20 or 100. Sometimes, UIDs are unsigned 16-bit numbers, ranging from 0 to 65535.

Unix keeps the translation between usernames and UIDs in the file **/etc/passwd.** Each user's UID is stored after that user's encrypted password. For example,

rachel:eH5/.mj7NB3dx:181:100:Rachel Cohen:/u/rachel:/bin/csh

In this example, Rachel's username is **rachel** and her UID is **181.**

The UID is the actual number that the operating system uses to identify the user; usernames are provided merely as a convenience for humans. If two users are assigned the same UID, Unix views them as the same user, even if they have different usernames and passwords. Two users with the same UID can freely read and delete each others' files and can kill each others' programs. Giving two users the same UID is almost always a bad idea. (The one exception to this rule is log-ins used for the UUCP system. In this case, it is desirable to have multiple UUCP log-ins with different passwords and usernames, but all with the same UID.)

Groups and identifiers

Every Unix user also belongs to one or more groups. Like usernames and UIDs, groups have both group names and GIDs (group identification) numbers.

Unix groups group users together. (Got that?) The system administrator assigns each user to one or more groups when the user's account is created. Groups let the system administrator designate specific groups of users who are allowed to access specific files, directories or devices.

Each user belongs to a primary group that is stored in the **/etc/passwd** file. The GID of the user's primary group follows the user's UID. Consider, again, our **/etc/asswd** example:

rachel:eH5/.mj7NB3dx:181:100:Rachel Cohen:/u/rachel:/bin/csh

In this example, Rachel's primary group identification number is 100.

Groups provide a handy mechanism for treating a number of users in a certain way. For example, you might want to set up a group for a team of students working on a project so that students in the group, but nobody else, can read and modify the team's files. Groups can also be used to restrict access to sensitive information or specially licensed applications stored on the computer. For example, many Unix computers are set up so that only users who belong to the **kmem** group can examine the operating system's kernel memory. The **ingres** group is commonly used to prevent non-registered users from running the Ingres database program.

The **/etc/group** file contains the database that lists every group on your computer and its corresponding GID. Its format is similar to the format used by the **/etc/passwd** file.

Here is a sample **/etc/group** file that defines four groups: **wheel, uucp, vision** and **users:**

```
wheel:*:0:root,rachel
uucp:*:10:uucp
vision:*:101:keith,arlin
users:*:100:
```

The first line of this file defines the **wheel** group. The fields are explained in the chart below.

The /etc/group fields

Field	Contents
wheel	The group name
*	The group's 'password'
0	The group's GID
root, rachel	The list of users in the group

By convention, the **wheel** group is the list of all of the computer's system administrators (in this case, Rachel and the **root** user are the only members). The second line of this file defines the **uucp** group. The only member in the **uucp** group is the **uucp** user. The third line of this file defines the **vision** group. This group lists all of the users working on a vision project. The **users** group does not explicitly list any users; each user on the system is a member of the **users** group by virtue of their individual entries in the **/etc/passwd** file.

Groups are handled differently by System V Unix and Berkeley Unix.

Under System V Unix, a user can reside in only a single group at a time. To change your current group, you must use the **newgrp**(1) command. The **newgrp** command takes a single argument: the name of the group that you're attempting to change into. If the **newgrp** command succeeds, it spawns a subshell that has a different group GID:

```
$ newgrp news
$
```

Usually, you'll want to change only into groups in which you're already a member, that is, groups that have your username in the **/etc/group** file.

System V also allows you to change into a group of which you're not a member. For this purpose, System V uses the group password field of the **/etc/group** file. If you try to change into a group to which you're not a member, the **newgrp** command will prompt you for that group's password. If the password you type agrees with the password for the group stored in the **/etc/group** file, the **newgrp** command goes ahead and temporarily puts you into the group:

```
$ newgrp fiction
password: yates34
$
```

You're now free to exercise all of the rights and privileges of the **fiction** group.

The password in the **/etc/group** file is interpreted exactly like the passwords in the **/etc/password** file, including salts. However, most systems do not have a program to install or change the passwords.

One of the many enhancements that Berkeley made to the Unix operating system was to allow users to reside in more than one group at a time. When a user logs in to a Berkeley Unix system, the program **/bin/login** scans the entire **/etc/group** file and places the user into all of the groups to which that user belongs. Thus, Berkeley Unix has no need for the **newgrp** command. Indeed many versions of Berkeley Unix do not include it.

In addition to regular users, Unix comes with a number of special users that exist for administrative and accounting purposes. The most important of these users is **root,** the superuser.

Every Unix system comes with a special user in the **/etc/passwd** file with a UID of 0. This user is known as the superuser and is normally given the username **root.** The password for the **root** account is usually called simply the "**root** password."

The **root** account is used by the operating system itself to accomplish its basic functions, such as logging users in and out of the system, recording accounting information and managing input/output devices. For this reason, the superuser exerts nearly complete control over the operating system. All security checks are turned off for any program run by the **root** user.

The **root** account is not an account designed for the personal use of the system administrator. However, the Unix system administrator will frequently have to become the superuser (usually using the **su**(1) command, discussed later in this chapter) in order to perform various system administration tasks.

[Note: Under some versions of Unix, it is not even possible to log in as the superuser from the **login:** prompt. Anyone who wishes to have superuser privileges must first log in as himself or herself and then **su** to **root.** This feature makes it easier to keep track of who is using the **root** account because the **su** command logs who runs it and when. Even if your system allows you to log in directly as **root,** we recommend first logging into your own account and then using the **su** command.]

Any process that has an effective UID of 0 runs as the superuser, that is, any process with an effective UID of 0 runs without security checks and is allowed to do almost anything. Normal security checks and constraints are ignored for the superuser, although some systems audit and log all activity by the superuser. The username **root** is merely a convention; remember, Unix uses the UID, not the username, for all security checking inside the operating system.

Some of the things that the superuser can do include:

Process control:
- Change the *nice* value of any process.
- Send any signal to any process.
- Alter "hard limits" for maximum CPU time as well as maximum file, data segment, stack segment and core file sizes.
- Turn accounting on and off.
- Can bypass log-in restrictions prior to shutdown (on Berkeley systems).
- Become any other user on the system.

Device control:
- Access any working device.
- Shut down the computer.
- Set the date and time.
- Read or modify any memory location.

Network control:
- Run network services on "trusted" ports.
- Reconfigure the network.
- Put the network interface into "promiscuous mode" and examine all packets on the network (possible only with some kinds of network interfaces).

Filesystem control:
- Read, modify or delete any file or program on the system.
- Run any program. (If a program has a file mode of 000, **root** must set the execute bit of the program with the *chmod*(2) system call before the program can be run.)
- Change a disk's electronic label. (Usually stored on the first 16 blocks of a hard disk or floppy disk formatted with the Unix filesystem.)
- Mount and unmount filesystems.
- Add, remove or change user accounts.
- Enable or disable quotas and accounting.
- Use the *chroot*(2) system call, which changes a process's **root** directory.
- Write to the disk after it is "100 percent" full. (The Unix file system saves the last 10 percent of the disk for work space and the use of the superuser, in some versions.)

What the superuser can't do:

Despite all of the powers listed above, there are some things that the superuser can't do, including:
- Make a change to a filesystem that is mounted read-only. (However, the superuser can unmount a read-only filesystem and remount it read/write.)
- Write directly to a directory or create a hard link to a directory (although these operations are allowed on some Unix systems).
- Decrypt the passwords stored in the **/etc/passwd** file (although the superuser can modify the **/bin/login** and **su** system programs to record passwords when they are typed).
- Terminate a process that has entered a wait state inside the kernel, (although

See **SECURITY** *page s6*

SECURITY *from page s5*

: superuser can shut down the comput-.., effectively killing all processes).

Problem with the superuser

The superuser is the main security flaw in the Unix system. Because the superuser can do anything, once a person gains superuser privileges, for example, by learning the **root** password and logging in as **root,** that person can do virtually anything to the system. This explains why most attackers who break into Unix systems try to become superusers.

Most Unix security holes that have been discovered are the kind that allow regular users to obtain superuser privileges. Thus, most Unix security holes result in a catastrophic failure of the operating system's security mechanisms. In other words, once a flaw is discovered and exploited, the entire computer is compromised.

There are a number of techniques for minimizing the impact of such system compromises. Two excellent ones are:

- Store your files on removable media so that an attacker who gains superuser privileges will still not have access to critical files.
- Store your files encrypted. Being the superuser grants privileges only on the Unix system; it does not magically grant the mathematical prowess necessary to decrypt a well-coded file or the necessary clairvoyance to divine encryption keys.)

Other operating systems, most notably Multics, bypass the superuser flaw by compartmentalizing the many system privileges which Unix puts into the root user. Indeed, attempts to design a "secure" Unix (one that meets U.S. government definitions of trusted systems) have adopted this same strategy of dividing superuser privileges into many different categories. But attempts at compartmentalization often fail. For example, Digital's VAX/VMS operating system divides system privileges into many different groups. But many of these privileges can be used by a persistent person to establish the others: An attacker who achieves "physical I/O access" can modify the operating system's database to give himself any other privilege that he desires.

To minimize the danger of superuser penetration, many Unix systems use other special user accounts to execute system functions that require special privileges – for example to access certain files or directories – but that do not require superuser privileges. These special users are associated with particular system functions, rather than individual users.

One very common special user is **uucp.** UUCP is a system for transferring files and electronic mail between Unix computers connected by telephone. When one computer dials another computer, it must first log in. Instead of logging in as **root,** the remote computer logs in as **uucp.** Electronic mail that's awaiting transmission to the remote machine is stored in directories that are readable only by the **uucp** user so that other users on the computer can't read each other's personal mail.

Other common special users include **daemon,** which is often used for network utilities, and **ingres,** which is used for the Ingres database program.

From the point of view of system security, **/etc/passwd** is the Unix operating system's most important file. If you can alter the contents of **/etc/passwd,** you can change the password of any user or make yourself the superuser by changing your UID to 0.

The **/etc/group** file is also very important. If you can change the **/etc/group** file, you can add yourself to any group that you wish. Often, by adding yourself to the correct group, you can eventually gain access to the **/etc/passwd** file and thus achieve all superuser privileges.

Sometimes, it's necessary for one user to assume the identity of another. For example, you might sit down at a friend's terminal and want to access one of your protected files. Rather than forcing you to log your friend out and log yourself in, Unix gives you a way to change your user ID temporarily. It is called the **su**(1) command, short for "substitute user." **su** requires that you provide the password of the user to whom you are changing.

For example, to change yourself from **tim** to **john,** you might type:

```
% whoami
tim
% su john
password: fuzbaby
% whoami
john
%
```

You can now access **john's** files.

The **su** command changes only your process's effective UID; the real UID remains the same. Likewise, as **su** does not change your entry in the **/etc/wtmp** or the **/usr/adm/utmp** files, the output from the **finger** command will not change after you use the **su** command. Many programs, such as **mail,** will not work properly when used from within a **su** subshell.

The **su** command is also the gateway to the superuser account and the privileges that the account possesses. It is often necessary to become the superuser. For example, you may be the system administrator and need to kill a runaway process. Normally, even the system administrator is allowed to kill only his or her own processes. To kill a process belonging to someone else, you must become the superuser.

Typing **su** without a username tells Unix that you wish to turn yourself into the superuser. You will be prompted for a password. Typing the correct **root** password causes a shell to be run with a UID of 0. When you become the superuser, your prompt should change to the sharp sign (#) to remind you of your new powers. For example:

```
% /bin/su
password: k697dgf
# whoami
root
#
```

When using the **su** command to become the superuser, it is always advisable to

See SECURITY *page s8*

• Distribution • Light Manufacturing • Wholesale • General Accounting •

Looking for a *reliable* business software system?

Down To Earth Business Software
is...
- *Easy to learn*
- *Easy to use*
- *Easy on your budget!*

Down To Earth Business Software
is...
12 individual 4GL software packages available for most business computers:

•General Ledger • Accounts Receivable •
• Accounts Payable • Payroll • Inventory •
•Job Cost • Order Entry • Purchase Order •
•Bill of Materials • Sales Analysis •
•Report Writer • System Manager/Security •

D2E
Down To Earth Business Software

•Portable across VMS, ULTRIX, UNIX, TSX & MS-DOS•
•Source code available • Dealer inquiries invited•
Interested? Call today for more information!

DIGITAL INFORMATION SYSTEMS CORPORATION
1-800-FON-DISC
or 916-635-7300 fax 916-635-6549

digital™ Authorized Solution Provider
As a Digital Authorized Solution Provider we can provide the best of two worlds hardware from Digital Equipment Corporation, one of the world's leading computer companies, and application software and services from DISC

Circle **301** on Reader Service Card (Page 8)

Mac300 Series Shifts Mac Communication into High Gear.

User speed and productivity are the big winners in White Pine Software's introduction of its new VT300 series terminal emulators. Mac320, Mac330, and Mac340 optimize Macintosh communications with VAX and Unix machines to display text (Mac320), monochrome text and graphics (Mac330) and color, text and graphics (Mac340).

We've neatly parlayed a number of advances – like multiple sessions, easy-to-use interface, ReGIS and SIXEL emulation and TCP/IP support to give the user unprecedented speed and control in the exchange of information. For the Unix user, we support MacTCP Telnet connections and file transfers using FTP.

PICK A SESSION, ANY SESSION

Unlike other limited terminal emulators, the Mac300 family lets users simultaneously open as many sessions as they want to as many hosts as they want. It's as though you have multiple terminals on one screen. No other terminal emulator offers true multiple sessions.

MultiFinder compatibility and proven file transfer techniques further facilitate inter-computer exchange.

WORKS WITH SYSTEM 7.0, A/UX AND COMMUNICATIONS TOOLBOX

Mac320, Mac330 and Mac340 are compatible with the latest Macintosh operating systems and work with the most commonly used Macintosh monitors. They also take full advantage of Apple's extended keyboard features, including user definable keys, function keys and the numeric keyboard. Through the use of Communications Toolbox, the Mac300 series emulators support various serial and network connections, including DECnet connections.

A RICH BONUS: FREE COMMUNICATIONS SOFTWARE

Mac300 series emulators support a range of communications options. Each package comes complete with the necessary software for making LAT, TCP/IP, direct serial and modem connections, free of charge. Support for Telnet and FTP is also included free.

ORDER NOW FOR GREATER PRODUCTIVITY

If you're ready to move your Macintosh communications into high gear, contact White Pine Software, 40 Simon St., Suite 201, Nashua, NH 03060-3043. Telephone (603) 886-9050. Fax (603) 886-9051.

WHITE PINE SOFTWARE

Circle **302** on Reader Service Card (Page 8)

NETWORK *from page s7*

ix in the Office," published by Patricia old's Office Computing Group in Boston, said, "Providing a single set of administration tools for heterogeneous Unix systems, while it will require some work, will be the easy part. The hard part will be bringing in a large base of PCs, PC networks, as well as minis and mainframes. To do that, you'll need the support of a very large number of vendors or a consortium such as the Open Software Foundation."

One of the most promising developments in system administration tools for open systems is the work of the OSF. DME (Distributed Management Environment), the OSF's system administration offering, is expected to be announced by the end of this year but not delivered until 1994. It is a set of application services that includes many critical management functions as well as a framework that provides the building blocks needed to manage diverse systems.

The OSF is a consortium of vendors including IBM, DEC and Hewlett-Packard Co., but not including Sun. OSF products include OSF/1, a Unix-like operating system; Motif, a graphical user interface for Unix and OSF/1 environments; and DCE (Distributed Computing Environment), a framework supporting distributed computing and database applications.

In the summer of 1990, the OSF issued a request for technology for an open systems-based distributed management architecture and a set of accompanying management applications. The OSF defined the following management and system administration tasks for submission: ounting, backup and restore, license agement, notification services, object onitoring and control, printer services, software installation and distribution and user management.

Late last year, the consortium accepted technologies from Tivoli Systems, Hewlett-Packard, Groupe Bull, IBM, Gradient Technologies, MIT's Project Athena and Banyan Systems Inc.

The OSF's Gossels said, "DME is the first vendor-neutral platform for maintaining multivendor networks. The first beneficiaries will be system administrators who will get a consistent user interface for doing management. In the past they had to learn how to use the system administration tools for each computer line."

Gossels added that his organization believes acceptance will be strong. "We chose the best available technologies and combined them in ways that are simple, elegant and powerful. But only time will tell if we get the support from users and manufacturers."

Christopher Baum, senior analyst at Datapro Information Services Group in Delran, N.J., was more upbeat. "OSF has solved most of the technical problems. And with HP, DEC and IBM backing the standard, there's no question about acceptance. It's a foregone conclusion."

"For most people, a Unix-only solution is not of that much interest because most users don't have Unix-only problems."

– Jonathan Gossels, OSF

Gail Ferreira, enterprise management framework business manager at DEC, said Digital is committed to implementing DME within six months of its introduction.

"We're tracking what OSF is doing; we're watching it closely," she said. "But we're not necessarily waiting for DME. Our customers want open systems. We can achieve that with standards, or we can achieve that with vendors getting together. We'll be continually looking for the best ways to give the customers the open systems that they demand." DME 1.1 probably won't be ready until the beginning of 1994, but Baum suggested that users who need open systems administration tools today can buy products from companies with technologies that were accepted by the OSF for DME, including Hewlett-Packard's Overview network manager and Tivoli Systems' Wizdom.

One important question determining utimately how useful DME will be is whether Sun will configure its system administration products to interoperate with DME. Currently, the answer from Sun is equivocal. "So far, DME does not exist. Once it's released, we'll determine if it has the potential of garnering the installed base to make it worth our while to create systems that interoperate with it," said Sun's Ulbrich. "If DME gets customer support, we will definitely work to connect with it. But at present, nothing is certain."

However, many market watchers think the likelihood is high that Sun's workstations will interoperate with DME. "There has been a lot of work going on between HP and Sun that will bring their stuff closer together. Since HP technology is part of DME, it is extremely likely that Sun will eventually support DME. What's not clear yet is how and when that will happen," said Goulde from Patricia Seybold's Office Computing.

Although exact timetables and the extent of interoperability are unclear, few doubt that the second half of this decade will see the realization of large-scale open networks administered by a single set of tools. Network managers who are used to proprietary networks with strong administration tools will be demanding the same functionality on their open systems. And one of the nice things about the competition brought about by open systems is that what the user wants, the user tends to get.

"It's clear by the amount of work being done by hardware and software vendors, the computer industry believes users will want a single set of friendly and powerful system administration tools to manage networks of diverse systems," said The Standish Group's Johnson. "And it is just as clear that by the time those networks are in place, system administration tools to manage them will be available." ■

Larry Stevens is a free-lance writer based in western Massachusetts.

Superuser: Unix security flaw

SECURITY *from page s6*

type the command's full pathname, which is **/bin/su.** By typing the full pathname, you are assuring that you are actually running the real **/bin/su** command and not another command named "su" that happens to be in your search path. This is a very important way you can protect yourself (and the superuser password) from capture by a Trojan horse.

To exit the subshell, type exit or press Ctrl-D.

If you use the **su** command to change to another user while you are the superuser, you won't be prompted for the username of the user who you are changing yourself into. (This makes sense; as you're the superuser, you could just as easily change that user's password and then log in as that user.)

Using **su** to become the superuser is not a security hole. Any user who knows the superuser password could just as well log in as superuser; it is no easier to break in through **su.** In fact, **su** enhances security: Many Unix systems can be set up so that every **su** attempt is logged, with the date, time and user who typed the command. Examining these log files allows the system administrator to see who is exercising superuser privileges, as well as who shouldn't be.

On newer versions of Berkeley Unix, a user cannot **su** to the **root** account unless the user is a member of the process group **wheel** (which must be given the group ID of 0). Some versions of **su** additionally allow members of the **wheel** group to become the superuser by providing their own password instead of the superuser password. The advantage of this feature is that you don't need to tell the superuser's password to a user in order for them to have superuser access – you just have to put them into the **wheel** group. And you can take away their access simply by taking them out of the group.

Some versions of System V Unix require that users specifically be given permission to **su.** Different versions of System V Unix accomplish this in different ways; consult your own system's documentation for details.

Another way to restrict the **su** program is by making it executable only by a specific group and by placing in that group only the people who you want to be able to run the command.

The bad su log

All versions of the **su** command log bad **su** attempts on the console and in the **/usr/adm/messages** file. If you notice many bad attempts, it is an indication that one of your system's users is trying to gain unauthorized privileges. A single bad attempt, of course, might simply be a mistyped password or somebody wondering what the **su** command does.

You can quickly scan the **/usr/adm/messages** file for bad passwords with the **grep** command:

```
% grep BAD /usr/adm/messages
Jan 20 10:28:38 prose su: BAD SU rachel on /dev/tty01
%
```

In this example, rachel tried to **su** on Jan. 20 and failed.

Other users of su

On older versions of Unix, the **su** command was frequently used in the **/usr/lib/crontab** file to cause programs run by **cron** to be run under different user IDs. A line from a **crontab** file to run the UUCP clean program (which trims the log files in the UUCP directory) might have had the form:

```
0 4 * * * su uucp -c /usr/lib/uucp/uuclean
```

This use of **su** is now obsolete: Berkeley Unix now requires that the username be specified as the sixth argument on each line of the **crontab** file:

```
0 4 * * * uucp/usr/lib/uucp/uuclean
```

System V Unix allows every user to have his or her own **crontab** control file. Each file is given the username of the user for whom it is to be run; that is, **cron** commands to be run as **root** are placed in a file called **root,** while **cron** commands to be run as **uucp** are placed in a file called **uucp.** The files are kept in the directory **/usr/spool/cron/crontabs.**

Summary

Every account on your Unix system belongs to one or more groups. You can use groups to limit the privileges that each user has.

Your computer has a special account called **root,** which has complete control over the system. Be sure to limit who has access to the **root** account and routinely check the console and the file **/usr/adm** messages for bad **su** attempts.

For more information about Garfinkel and Spafford's "Practical Unix Security," contact O'Reilly & Associates Inc., 632 Petaluma Ave., Sebastopol, Calif. 95472, (800) 338-6887. ■

ADVERTISERS INDEX

Reader Service No.	Page
Digital Equipment (602) 991-8794	s2
301 DISC (800) FON-DISC	s6
302 White Pine Software (603) 886-9050	s6

This index is provided as a service to our readers. DIGITAL News is not responsible for inaccurate listings.