Superuser

Almost 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 the identity 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: nearly all security restrictions are bypassed for any program that is run by the *root* user, and more of the checks and warnings are turned off.

What the Superuser Can Do

Any process that has an effective UID of 0 runs as the superuser, i.e. any process with a 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 most systems do audit and log some of the superuser's actions.

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 and auditing on and off
- Bypass login restrictions prior to shutdown
- Change root's process UID to that of any other user on the system
- Logout all users and prevent new logins

Device Control

- Access any working device
- Shut down or reboot the computer
- Set the date and time
- Read or modify any memory location
- Create new devices

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

File System Control

- Read, modify, or delete any file or program on the system
- Run any program

- Change a disk's electronic label
- Mount and unmount file systems
- Add, remove, or change user accounts
- Enable or disable quotas and accounting
- Use the **chroot()** system call, which changes a process's view of the file system root directory
- Write to the disk after it is "100 percent" full.

What the Superuser Cannot Do

Despite all of the powers listed in the previous section, there are some things that the superuser cannot do including:

- Make a change to a file system that is mounted read-only
- Unmount a file system that contains open files, or one in which some running process has set as its current directory
- Write directly to a directory or create a hard link to a directory
- Decrypt the passwords stored in the shadow password file although the superuser can modify the **/bin/login** and **su** system programs to record passwords when they are typed
- The superuser can also use the **passwd** command to change the password of any account
- Terminate a process that has entered a wait state inside the kernel although the superuser can shut down the computer effectively killing all processes

Problems with the Superuser

The superuser is the main security weakness in the UNIX operating system. Because the superuser can do anything, after a person gains superuser privileges, e.g. 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 the superuser.

Most UNIX security holes that have been discovered are of the kind that allow regular users the ability to obtain superuser privileges. Thus, most UNIX security holes result in a catastrophic bypass of the operating system's security mechanisms. After a flaw is discovered and exploited, the entire computer is compromised.

There are a number of techniques for minimizing the impact of each system compromise including:

- Storing sensitive files on removable media and mounting the media only when you need to access the files. An attacker who gains superuser privileges while the media are unmounted will not have access to critical files,
- Encrypting your files. 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. Best practice is to encrypt with a passphrase other than your login password, which an attacker might capture.
- Mounting disks read-only when possible
- Taking advantage of file system features like immutable and append-only files if your system supports them

- Keeping your backups of the system current.
- Restricting superuser login to the console.

Restrictions on the Superuser

Since the superuser account is occasionally compromised, e.g. by someone sharing the superuser password with a friend, there have been numerous attempts to limit the availability and the power of the UNIX superuser account

Secure Terminals: Limiting Where the Superuser Can Log In

Solaris allows you to configure certain terminals so that users cannot log in as the superuser from the login: prompt. Anyone who wished to have superuser privileges must first log in as herself and then **su** to *root* or use **sudo** to execute commands. This feature makes tracking who is using the root account easier because the **su** and **sudo** commands log the username of the person who runs it and the time that it was run. Solaris also requires that the *root* user's password be provided when booting into single-user mode if the console is not listed as being secure.

Secure consoles add to the overall system security because they force people to know their own password before they can gain superuser access to the system. Network virtual terminals should not be listed as secure to prevent users from logging into the *root* account remotely using **telnet** or **ssh**. Of course **telnet** should be disabled!

Solaris is configured so that the superuser can log in to the *root* account on the system console but not on any other terminal.