Next Previous Contents

14. Security

There are many kinds of security mechanism supported by the Linux kernel.

14.1 seccomp

Since it is rather unknown, and seems undocumented so far, let me describe the seccomp feature.

One can set up a sandbox for a process, where the process can use only a restricted set of system calls, and is killed as soon as it tries to do anything else. (I think that is too harsh. Returning ENOSYS for every system call that is not in the allowed set seems better.)

It was introduced in 2.6.12 by Andrea Arcangeli with a procfs interface. An

```
% echo 1 > /proc/self/seccomp
```

would put a process into seccomp mode 1 (the only nonzero mode implemented so far). In 2.6.23 the interface was changed to a one via `prctl()`. The equivalent action is now

```
prctl(PR_SET_SECCOMP, 1, 0, 0, 0);
```

Afterwards, this process cannot use any system call other than read/write/exit/sigreturn. (In particular, `exit(n)` will fail since that uses `exit_group()`. One needs `syscall(SYS_exit, n)`.)

The purpose of this mode is to set up a process for an untrusted computational task. One might use it to securely decompress movies or audio files or jpegs with little overhead. The only interface between process and the rest of the system are its open file descriptors, and signals. (Note that glibc uses several system calls for the first printf, so if stdout is to be used with the libc printf, then at least one printf is needed before enabling seccomp.)

There is no way for a process to test whether it is running in seccomp mode without killing itself if it is.

Next Previous Contents