### Assignment 1: Unix access control

In Unix, every process has a real user id (*ruid*), an effective user id (*euid*), and a saved user id (*suid*). Processes with an euid of 0 have special root privileges.

1. If a process with user id *n* forks to create another process, what user id does the new process have? (*Hint:* it's the same answer for euid, ruid, and suid.)
2. If a process with euid *n* makes a setuid system call, what possible euids can the process run with after the call, in each of the following situations:
   1. Before: *euid = n > 0*, saved user id *suid=m* and real user id *ruid = m*. After:?
   2. Before: *n=0* After:?
3. In qmail, most modules run under separate user ids. Similarly, each Android application runs in a separate process using a separate user id. From a security standpoint, what is the advantage of assigning separate uids instead of using the same uid for all? Explain.
4. The Android zygote process that creates new processes runs as root. After forking to create a new process, setuid is normally called. Explain what uid the new process has initially and why it is important to call setuid? What security purpose does this serve?
5. When a Unix user wishes to change her password, she uses the passwd program. The Unix password file is usually public readable but (for obvious reasons) can only be written by processes with root privileges.
   1. How should the *setuid* bit be set on this passwd program? Explain how this lets a user change her password.
   2. Why does this make it important to write the passwd program source code carefully?