Homework 3

Due: November 20, 2014 **Points:** 100

Questions

1. (*20 points*) Discuss controls that would prevent Dennis Ritchie's bacterium (see Section 22.5.1) from absorbing all system resources and causing a system crash.

2. (*20 points*) Millen's report on Operating System Verification report does not give an initial state for the variables MEM(b, i), MAR(i), and the supervisor variables introduced on page 22. Provide initial values for all of these variables that assure that the Access Control Policy (R1) defined on page 26 is satisfied in the initial state. (Note that all of the processes, p must exist in the initial state since there is no operation that creates a new process.)

3. (*20 points*) On page 33 of Millen's report, consider the "new" supervisor variable X and the "new" supervisor operations READ(i) and WRITE(i). Show that there is a flow violation associated with the calls to these operations for all *reasonable* assignments of levels to X. Don't stress out over this, as there are not that many reasonable levels; X can be assigned the maximum lebel "syshi", the minimum level"public", PAL(P1), or BAL(P1).

4. (*40 points*) The program *setdate* runs setuid to *root* on a Linux system, so whenever it is executed, it runs with *root* privileges and not with the privileges of the user executing the program. Its function is to change the date of the system. Please analyze the robustness and security of this program by looking at the source code (available from the class web page). How would someone obtain excess privileges using this program? What input would someone need to supply input to cause the program to crash, or do something other than changing the date?

Extra Credit

1. (*10 points*) The C shell does not treat the **IFS** variable as a special variable. (That is, the C shell separates arguments to commands by white spaces; this behavior is built in and cannot be changed.) How might this affect the *loadmodule* exploitation discussed in section 23.2.8?

2. (*20 points*) Prove that the RSA decryption algorithm does in fact decrypt a message enciphered using the RSA encipherment algorithm.
Hint: Let m be the message and n be the modulus. First, prove the claim when m and n are relatively prime. Then prove the claim when m and n are not relatively prime.