CSci 5271: Introduction to Computer Security

Exercise Set 2 **due: October 9, 2012**

Ground Rules. You may choose to complete these exercises in a group of up to three students. Each group should turn in **one** copy with the names of all group members on it. You may use any source you can find to help with this assignment but you **must** explicitly reference any source you use besides the lecture notes or textbook.

1. Equal Error Rates and Passwords. Many biometric authentication schemes produce a "confidence" value that allows a tradeoff between "false positive" and "false negative" errors. Password schemes are not typically considered in this light. List some reasons why you think this might be. We could change the way we check passwords to produce a confidence value; for example, the edit distance between a login attempt and the stored password. What are the (security) challenges of this approach compared with the standard use of passwords? (Feel free to generalize to other biometrics here) How would you measure the EER of a password system using this approach? (Would your measurement be meaningful?)

2. Access control without hardware support. Alice is a developer for a toy company. One day her boss Cindy rushes up to her desk excitedly and says "we are going to develop a toy computer with an operating system and everything." Alice is really excited about the prospect of developing an operating system until she finds out that Cindy has already purchased processors that have no access control mechanisms at all - neither a supervisor bit nor a MMU. On the plus side, they are really fast and she has tons of RAM. Alice thinks for a bit longer and decides she can solve this problem in a pretty straightforward way. Sketch out her solution, in enough details to convince a fellow student it will be secure.

3. Sharing files in Unix. Alice wants to be able to share read and write access to some of her files (on a unix system) with dynamically changing sets of users. Since she is not root, she can't just construct new groups for each file, nor can she turn on the optional ACL feature available on some Linux systems. So she decides to write setuid programs that will implement ACLs for her friends. Alice designs two setuid, world-executable programs, `alice-write` and `alice-read` (e.g. programs that anyone can run as `alice`) that work as follows:

- `alice-write in out` first checks a permission file written by Alice to make sure that the ruid of the process (the calling user) is allowed to write to the file `out`. If so, then the program reads the file `in` and writes it over `out`.
- `alice-read in out` first checks a permission file written by Alice to make sure that the calling user is allowed to read the file `in`. If so, the the program reads `in` and writes it to the file `out`.

Assume Alice has been careful in her implementation, i.e., there are no buffer overflows in `alice-read` and `alice-write`, the permission file is properly protected (uniquely named in the program and set to permission 0400), the programs accept only file paths listed in the permissions file, and permissions on Alice's files are preserved. What are some of the potential security problems with this approach? (e.g. suppose Bob can read and write some of Alice's files but not others;

can he use `alice-write` and `alice-read` to gain access to files he shouldn't? Are there potential attacks that could allow third parties to read/write Alice's files?)

You can read more about the setuid functions in the paper "Setuid Demystified" by Chen, Wagner, and Dean, which is available for download at

`http://www.cs.berkeley.edu/~daw/papers/setuid-usenix02.ps` .

Suggest ways to change the interface and/or implementation of `alice-write` and `alice-read` to avoid your attacks.

4. Multilevel-secure file sharing in Unix. Bob is setting up a Unix that does not support POSIX ACLs. His boss has told him that they will be using a multi-level classification system with three ranks: public $<$ private $<$ management, and one specialized compartment, `personnel`. Every user will hold a clearance according to this system.

(a) Suppose Alice has current clearance (private, $\emptyset$). Draw the lattice of classifications in this system (there are 6 classifications) and mark with an "`r`" each classification that Alice should be able to read under the BLP policy and a "`w`" each classification that Alice should be able to write to under the BLP policy.

(b) Bob proposes to solve the problem of enforcing BLP access control as follows. For every classification C, he makes two groups, $C-$`r` and $C-$`w`. If a user is cleared to *write* to files of classification C, she will be added to group $C-$`w`, and if she is cleared to *read* files classified C she will be added to group $C-$`r` (if she is neither cleared to read or write to C, she will not be added to either group). If he had ACLs, this would be sufficient: a file with clearance C could grant write access to the group $C-$`w` and read access to the group $C-$`r`.

Since Bob doesn't have ACLs, he gets creative: for every file `f` that needs BLP access controls, he creates a directory `f-dir` that is owned by the "write" group for the `f`'s classification, and a file `f-dir/f-data` that is owned by the "read" group for `f`'s classification. So for example, if the file `pay-db` is classified (management, `personnel`) there will be a directory `pay-db-dir` owned by group `mgmt-pers-w` with permissions 0075, and a file `pay-db-dir/pay-db-data` owned by group `mgmt-pers-r` with permissions 0040. In this way, a user who needs to write the file can remove the old one (using write permission on the directory) and create a new one. Bob also creates a separate process that monitors directories and changes permissions appropriately (so that after the user writes a new `pay-db-dir/pay-db-data` file it will be changed over to the correct group ownership.).

What are some of the problems with this approach? (i.e., are there conditions under which classified data is leaked to uncleared users? when and how? Can cleared users be prevented from reading classified documents? how?) Suggest a better solution without using ACLs.