Name: _____ UNI: _____

# COMS W4187: Security Architecture and Engineering
March 2007

## Rules

- Remember to write your name and UNI on the blue exam book.
- **Important: also write your name on this paper**
- You must turn in *both* the exam sheet and the blue book
- Books and notes are allowed during this examination; computers are not permitted.
- This is a time-limited test. All papers must be turned in 75 minutes after the beginning of the test.
- The total points add up to 85.
- Good luck, and may the Force be with you.

| | | |
|---|---|---|
| 1 | | 15 |
| 2 | | 15 |
| 3 | | 10 |
| 4 | | 20 |
| 5 | | 10 |
| 6 | | 15 |
| Total | | 85 |

## Questions

**1.** (15 points) In class, all discussions of mandatory access control assumed an access control list protection scheme. Could Unix-style user/group/other permissions be used with mandatory access control? Explain.

Yes, it's possible. The essence of mandatory access control is that the owner of the file is not allowed to change its permissions. It doesn't matter what sorts of permissions they are.

**2.** (15 points) We discussed cryptographic key storage in class. To avoid some of the problems, a user proposes to keep keys for his ordinary, single-user laptop on a USB flash disk. He'll insert the USB flash disk when necessary, perform the cryptographic operation, and remove the flash disk. However, virtually all flash disks are formatted as a "FAT" file system, which has no access control at all. Does this property affect his idea? Why or why not?

On a multi-user system, this might matter, since other users would then have access to the files containing the keys. On a single-user machine, it probably doesn't matter. However. . . We can hypothesize a system where access to keys is mediated by some privileged process, to prevent access by viruses, Trojan horses, and the like. In that case, the lack of access control is indeed a problem.

**3.** (10 points) Some servers, such as FTP, handle Unix-style filename expansion. That is, they accept requests for things like `*.c`. To do the expansion, someone has suggested just asking the shell. In this case, for example, the program would pass

```
echo *.c
```

to the shell and let it do the expansion. If not done carefully, though, this is a very dangerous technique. Explain the dangers and what could be done to prevent them while still using this technique.

The shell interprets many other metacharacters besides those used for filename expansion (sometimes known as "globbing"). You'd have to filter the input to make sure there were no other dangerous characters.

**4.** (20 points) There is a network file transfer daemon that maintains a queue of pending and past requests. It uses three directories, with the following ownership and permission:

| *Name* | *User*:*Group* | *Modes* |
|---|---|---|
| pending | trans:daemon | `rwx,rx,x` |
| done | trans:daemon | `rwx,rx,x` |
| data | trans:daemon | `rwx,-,-` |

Files in the "pending" directory are control files that describe data to be transferred; when a transfer is complete, the control file is copied to the "done" directory. The actual data is in the "data" directory. Control and data files are also in user:group *trans:daemon*; the former are mode `rw,rw,-` and the latter are mode `rw,-,-`.

a. (10 points) The program that actually transfers files has to be able to read the control file directory, read individual control files, read the associated data files, and move the control files to the "done" directory. What privileges must this program run with?

This program has to be able to write to the "done" directory. Only user *trans* can do that. That user can read the directory and the data files, too.

b. (5 points) The program that gives the queue status needs to be able to read the control file directory and individual control files. It does not need to read data files. What privileges must this program run with?

Group *daemon* can read the necessary files. So can user *trans* — but see below.

c. (5 points) Why is it a good idea for the queue status program to have fewer privileges than the transfer program?

This is the principle of least privilege — programs shouldn't have permissions they don't need.

**5.** (10 points) The following code fragment occurs in a setUID root program. There are two security flaws in it. Identify each one and describe — in words, not code — what should be done instead. Note: there are *not* three flaws.

```
char buf[100], path[200];
int fd;

gets(buf);
strcpy(path, "/var/spool/");
strcat(path, buf);
fd = open(path, O_RDWR);
...
```

The call to `gets` is vulnerable to buffer overflows; it should use `fgets`. The input string is not filtered before the `open()` call, which would allow other files to be read instead. One of the mechanisms discussed in class for opening files with the user's permissions should be used instead. The `strcopy()` and `strcat()` are not erroneous; given the size of `/var/spool/` and `buf`, it isn't possible to overflow `path`.

**6.** (15 points) A brokerage firm wishes to use very strong authentication on its web site to protect its (very rich) customers' assets. They plan to do this by giving each customer an iris scanner to attach to their home computers.

a. (10 points) Will this form of biometric authentication help secure the site? Why or why not?
As discussed in class, remote biometrics aren't very useful.

b. (5 points) Once they've solved the authentication problem, what else should they look at?
Most security problems are due to buggy code, not authentication failures.