Sample Final Answers

These are sample questions that are very similar to the ones I will ask on the midterm. I expect the final will be approximately the same length.

1. Consider a system that used the Bell-LaPadula model to enforce confidentiality and the Biba model to enforce integrity.

   (a) If the security classes were the same as integrity classes, what objects could a given process (with some security class that also served as its integrity class) access?

   (b) Why is this scheme not used in practice?

   Answer:

   (a) The process could access exactly those objects in its security/integrity class. Reading an object in another class would violate either the simple security property (if the object's security level dominates the process' security level) or the integrity *-property (if the process' security level, which is the same as its integrity level, dominates the object's security level, which is the same as its integrity level). Similarly, writing an object in another class would violate either the simple integrity property (if the object's security level dominates the process? security level) or the *-property (if the process' security level, which is the same as its integrity level, dominates the object's security level, which is the same as its integrity level).

   (b) In practice, this limits the sharing that processes can do. As most computing today requires communication and shared resources, the above scheme would inhibit effective use of computers.

2. Define each of the following terms in one short sentence:

   (a) public key cryptosystem

   (b) challenge-response

   (c) computer worm

   (d) end-to-end encryption

   Answer:

   (a) A public key cryptosystem is a cryptosystem with two keys, one of which is known to everyone (the *public key*) and the other of which is known only to one person (the *private key*).

   (b) A challenge-response refers to a mechanism in which the client is challenged to demonstrate knowledge of a secret to a server. The client's response must match the response computed by the computer.

   (c) A computer worm is a program that copies itself onto another system.

   (d) End-to-end encryption occurs when a message is enciphered, and sent to its destination, and then deciphered at its destination. It remains enciphered as it transits the network, leading to the term "end-to-end encryption."

3. Show how ACLs and C-Lists are derived from an access control matrix.

   Answer: ACLs correspond to the columns of an access control matrix, and C-Lists correspond to the rows of an access control matrix.

4. Discuss the revocation problem with respect to access control lists and capabilities. How might one efficiently implement a command to revoke access to an object by one particular user?

   Answer: Suppose we wanted to revoke subject s's access rights r to a file f. If the system used access control lists, one would revoke the access by going to f's ACL and deleting s's rights r. If the system used capability

lists, one would revoke the access by going to s's capability list and remove the capability that gives s the r rights over f.

With ACLs, it is trivial to remove all rights to a given object from any subject. Simply delete that subject's entry from the ACL. With C-lists, it is much more difficult. For example, suppose we want to remove all users' rights to read a file f. We need traverse only one ACL, but we will need to traverse every process' C-List to see if that process has read rights over f. So we can create a global object table. Each entry in this table contains a pointer to either an object or to another table entry. Then we use entries from this table to identify objects in capabilities. Given this structure, revocation simply involves deleting the entry in the global object table. Thus, the hard part is managing the table in such a way that a process can dot he revocations it wants.

5. What is a certificate? What is it used for?

   Answer: A certificate is a construct that binds the representation of the identity of a subject (such as the subject's name) to a public key, and that is digitally signed by the entity that issued it. It is used to pass public keys through an infrastructure in such a way that there is some assurance to whom the public key belongs.

6. The following routine reads a file name from the standard input and returns its protection mode. It treats the argument as a file name, and returns the protection mode of the file as a short integer. Identify three non-robust features of this routine, and state how to fix them.

```
/* return protection mode of the named file */
short int protmode(void)
{
    struct stat stbuf;
    char inbuf[100];

    gets(&inbuf);
    stat(inbuf, &stbuf);
    return(stbuf.st_mode&0777);
}
```

   Answer:

   (a) The function *gets* does not check the length of the input. Use *fgets* instead.

   (b) The return value of *gets* (or *fgets*) is not checked. If the user types an end-of-file, *inbuf* will contain whatever data happens to lie in the area allocated for it. The right fix is to replace the *gets* line with:

```
if (fgets(&inbuf, 100, stdin) == NULL)
    return(-1);
```

   where -1 is an error code (this includes the fix from part a, above).

   (c) The return value of *stat* is not checked. The result will be that *stbuf* contains whatever data happens to lie in the area allocated for it. The right fix is to replace the *stat* line with:

```
if (stat(inbuf, &stbuf) == -1){
    perror(inbuf);
    return(-1);
}
```

   where -1 is an error code.