# My Answers to the Sample Midterm

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

1. Why is a precise statement of security requirements critical to the determination of whether a given system is secure?

   *Answer*: A precise statement of security requirements is critical to the determination of whether a given system is secure because the statement of security requirements underlies the system security policy, which in turn defines what "secure" means for that site. Without such a statement, one cannot determine whether the system is secure because one does not know what "secure" means.

2. This function's purpose is to copy a string from one buffer to another. It is not robust. Find the problems and say how to fix them. Note that the passing of pointers here is defined in the specification of the interface, and so cannot be changed.

   ```
   void mystrcpy(char *s, char *t)
   {
   while(*t != '\0')
   *s++ = *t++;
   *t = '\0';
   }
   ```

   *Answer*: There are two problems. Neither parameter is sanity-checked. Both should be compared to **NULL**.

3. Which of the following demonstrate violations of the principle of least privilege? Please justify your answer.

   (a) The Linux *root* account?

   (b) A user whose function is to maintain and install system software. This user has access to the source files and directories, access to only those programs needed to build and maintain software, and can copy executables into system directories for other users. This user has no other special privileges.

   *Answer*:

   (a) As any process running as *root* can do anything, whether or not it be related to the particular task that the process is to perform, the *root* account demonstrates a violation of the privilege of least principle.

   (b) In this case, the user can only perform system tasks related to the installation of software. Hence this does not demonstrate a violation of the principle of least privilege.

4. Into which category or categories of the Program Analysis classification do the following fall?

   (a) Buffer overflow causing a return into the stack?

   (b) Allowing an ordinary user to alter the password file?

   (c) Simultaneous writes to a shared database?

   (d) Reading a UNIX file by directly accessing the raw device and reading first the superblock, then the file's inode, and finally the file's data blocks?

   *Answer*: Answers other than these are possible. Here are mine.

(a) Buffer overflow causing a return into the stack is an example of improper validation. It arises from not checking the length of the string being loaded into the buffer. For the specific case in this problem (return into the stack), one could also argue that it is an example of improper choice of operation or operand, because the operand (the string) is meant to be data, but is in reality instructions. Further, you can view the storage of the return address in the stack as allowing the program to alter something (the return address) that should not change, making this an example of improper change.

(b) Allowing an ordinary user to alter the password file is an example of improperly setting the initial protection domain, because the user should not be able to alter the password file. If the user were to change the password file, this could also be viewed as an example of improper change (but the problem said "*allowing* an ordinary user to alter" and not "an ordinary user altering".

(c) Two processes updating a database simultaneously can cause inconsistencies. As they must synchronize themselves, failing to do so is improper synchronization, specifically an improper sequencing of the writes. You can also argue that intermingling the rights enables operations that should be indivisible (the writes) are not, and hence this is an example of improper indivisibility.

(d) Accessing a file by reading the raw disk bypasses the abstraction of "file" in the UNIX operating system. Hence this is an improper isolation of implementation detail.

5. Represent a security compartment label using the notation

$$( \textit{security level} \, ; \, \textit{set of categories} \, )$$

Can a user cleared for $( \textit{secret} \, ; \{ \textit{dog} , \textit{cat} , \textit{pig} \})$ have read or write access (or both) to documents classified in each of the following ways under the military security model?

(a) $( \textit{top secret} \, ; \{ \textit{dog} \})$
(b) $( \textit{secret} \, ; \{ \textit{dog} \})$
(c) $( \textit{secret} \, ; \{ \textit{dog} , \textit{cow} \})$
(d) $( \textit{secret} \, ; \{ \textit{moose} \})$
(e) $( \textit{confidential} \, ; \{ \textit{dog} , \textit{pig} , \textit{cat} \})$

*Answer*:

(a) No read (as $\textit{top secret} > \textit{secret}$); no write (as $\{ \textit{dog} , \textit{cat} , \textit{pig} \} \not\subseteq \{ \textit{dog} \}$)
(b) Read (as $\textit{secret} = \textit{secret}$ and $\{ \textit{dog} \} \subseteq \{ \textit{dog} , \textit{cat} , \textit{pig} \}$); no write (as $\{ \textit{dog} , \textit{cat} , \textit{pig} \} \not\subseteq \{ \textit{dog} \}$)
(c) No read (as $\{ \textit{dog} , \textit{cow} \} \not\subseteq \{ \textit{dog} , \textit{cat} , \textit{pig} \}$); no write (as $\{ \textit{dog} , \textit{cat} , \textit{pig} \} \not\subseteq \{ \textit{dog} , \textit{cow} \}$)
(d) No read (as $\{ \textit{moose} \} \not\subseteq \{ \textit{dog} , \textit{cat} , \textit{pig} \}$); no write (as $\{ \textit{dog} , \textit{cat} , \textit{pig} \} \not\subseteq \{ \textit{dog} , \textit{moose} \}$)
(e) Read (as $\textit{confidential} < \textit{secret}$ and $\{ \textit{dog} , \textit{cat} , \textit{pig} \} \subseteq \{ \textit{dog} , \textit{cat} , \textit{pig} \}$); no write (as $\textit{confidential} < \textit{secret}$ )