Answers to Sample Midterm Questions

1. I found two. First, the target string that *s* points to is never **NUL**-terminated; the line after the **while** loop should read

```
*s = '\0';
```

Second, neither parameter is checked to ensure it is not **NULL**. The fix for this is to insert the line

```
if (s == NULL || t == NULL) return;
```

just before the **while** loop.

2. 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.

3. Adding security features that expand the protection domain of some users beyond what is necessary to perform their functions violates the Principle of Least Privilege. If the additions restrict users further, there is not such violation. However, in that case, the unaugmented system did not adequately enforce this privilege.

If the added services grant privileges or access by default, they violate the Principle of Fail-Safe Defaults. If the additions disable accesses granted by default, the added features improve the system security (and, again, the base system violated this privilege initially).

Adding security services adds to the complexity of a system because the extra features are typically layered onto existing services that the system supplies. Because these added services must be considered in conjunction with the system, rather than as an integral part of the system, they violate the Principle of Economy of Mechanism. If the system is redesigned so that the added security services are integrated into the system completely, then there is effectively a new system with the services designed in. This does not violate the principle.

Adding security features may improve the mediation, so more accesses are checked than were the features not present. So in all likelihood, the added features improve compliance to the Principle of Complete Mediation. Of course, if they allow processes to avoid having accesses checked, they violate this principle. What is critical is that the security add-ons will increase the number of interfaces between the kernel and the security checking, so these low-level accesses (between the kernel and the added security features) must be carefully checked.

If the added security features do not depend on secrecy for their effectiveness, the Principle of Open Design is met. Otherwise, it is not. (Remember, a cryptographic key or a password may be kept secret; at issue here is the design of the mechanism, not the keys!)

Depending on the added functionality, the added security features may add an additional layer of checking. In this case, they improve adherence to the Principle of Separation of Privilege. If, on the other hand, they combine different checks into a single check (so one or more of the checks are bypassed), they weaken adherence to this privilege.

Security add-ons may obscure channels along which communication might occur, or can mediate such communications. As such, they can hinder the sharing of a resource, or mediate it. In this case, they aid in the system's adhering to the Principle of Least Common Mechanism. But if the security add-on is itself a resource that is shared, it introduces an additional shared resource that can be used for process interaction, thus violating the principle.

Adding security features requires careful integration into the user's model of how the system works. Note this is not necessarily the model used to design the system; it is the way that the user views the system. If the integration is not complete, the user may have to reconcile two different models, causing confusion. In this case, using the added security features is more difficult than not using them, a clear violation of the Principle of Psychological Acceptability. If, on the other hand, the added security features fit into the user's model of the system, using the added security features will not be more difficult than not using them, in which case the principle is upheld.

4. During the Flaw Hypothesis stage, the analyst uses the vulnerabilities models to draw parallels between the system under analysis and systems with known vulnerabilities. From these, he or she forms hypotheses of vulnerabilities in the system under analysis. The models may also be used in the Information Gathering phase (to highlight which areas of the system to focus upon) and the Flaw Generalization phase (to suggest

ways to generalize the problem), but neither of these uses is critical. Indeed, one can argue that the latter two uses limit the analyst to the view of the systems studied in the models. In any case, the vulnerabilities models are merely guides. They should not replace creative analysis.

5. 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. 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.

   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.

   c. Two processes updating a database simultaneously can cause inconsistencies. As they must synchronize themselves, failing to do so is improper synchronization.

   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.