Week 1 Quiz

1. What does "defensive programming" mean?

   - When you test your program, use the input that you expect the user (or the environment) to supply.
   - Program so that, if some assumption about the input or environment is wrong, the program will detect that and do something reasonable.
   - Assume some of the code you write will be compiled and executed incorrectly, so the program must run correctly even if that happens.
   - Write your program carefully so it has no syntax errors.

   Correct Defensive programming is like defensive driving -- assume something in the environment or input will not be what your program expects, and be prepared to handle it in some way.

2. Which of the following statements about a setuid program is/are true?

   - A setuid program can only access resources available to everyone.
   - A setuid program executes with the privileges of the user executing it.
   - A setuid program gives the user executing it additional privileges during the execution.
   - A setuid program always executes with the privileges of root.

3. Complete the sentence: A characteristic of _______ programming is that it handles internal errors gracefully.

   - Erroneous
   - Paranoid
   - Robust
   - Fragile

   Correct Robust programming is a style of programming that prevents abnormal termination or unexpected actions. In other words, if the program does what you want it to do, it's robust.

4. A program needs to obtain a set of web pages over the network, as well as perform other network functions. Which of these best describes how this should be done?

   - Put both the network access code and the web page processing code into a single module, and have the program call this encapsulation whenever needed.
   - Put the network access code in a separate module or file, and have the program call this encapsulation whenever needed.

- Put the network access code into the program wherever the network is accessed; this is efficient and allows you to change the code at the point in the program where it is used.
- Put both the network access code and the error-checking code at the beginning of the program, so the rest of the program will not have to check the future accesses; if bad, this will be caught before anything is done.

Correct Network access should not be scattered throughout your program. They should be in one, or a small set of, modules, and each of those modules should have appropriate checking code so if something goes wrong, the module will detect it. That relieves the higher levels that call these modules of the burden of checking the network parameters and conditions.

5. What is the LAND attack?

- An attack in which the attacker sends packets with the same source and destination address.
- An attack in which there is a path between the attackers and the target system that does not cross water.
- An attack in which the attackers flood the target system with packets more quickly than the target can process it, overwhelming the target.
- The first stage of a multi-stage attack that must succeed for the attack to be successful.

Correct The attack involves sending a spoofed TCP SYN packet (connection initiation) with the target host's IP address to an open port as both source and destination. This causes the machine to reply to itself continuously. It is, however, distinct from the TCP SYN Flood vulnerability.

6. Which of the following is true?

- Your program can be robust and secure even if the infrastructure is not.
- The infrastructure underlying your program (compilers, linkers, and so forth) have been shown to be secure, so any problems that arise must have come from your program.
- A non-robust, non-secure infrastructure can corrupt your program even if the code you write is completely robust and secure.
- Security problems arise from the user input, so if that is rigorously checked and bad inputs handled, your program will be robust.

Correct It is like a chain is only as strong as its weakest link. In programming, if the infrastructure is not secure, if the Libraries, the compiler, the link or loader, and so forth, the network services you rely on, aren't secure, or aren't robust, then its very hard to build a program that's robust for a service that's robust on top of them.

7. Consider the environment in which a program executes. You need to be concerned about the differences between that environment and the one in which the program was developed because:

   - The system may fail, causing the program to fail.
   - The assumptions under which the program was developed may not hold in the new environment.
   - The configuration of the program in the new environment must present the program with the same environment is that in which it was developed.
   - The users may enter incorrect input, causing the program to take unexpected actions.

   Correct As an example you might be able to use a library that was crafted to do one thing and it did it very, very well, but you use the library for a different purpose or in another environment. You must be aware of and document your assumptions as you move your program into another environment, those assumptions may no longer be valid in this new environment. It's imperative you document those assumptions, so when people do run your program in their environments, they can see, what could go wrong.

8. A network server runs with no privileges. Why is the robustness and security of this server of concern?

   - The server can send invalid data to the client, causing the client to malfunction.
   - The server can delay responding to the client, causing a time-out and appearing to be unavailable.
   - The server can refuse to answer the client, causing a denial of service.
   - The server is on a different system than the client, so the client is getting access to the server system.

   Correct In this case we are concerned about the security of the server and this system it runs on, rather than how the client handles errors.

9. Which of the following is the most correct?

   - Software composed of non-secure components will usually be secure.
   - Software composed of secure components will never be secure.
   - Software composed of non-secure components will usually be non-secure.
   - Software composed of secure components will always be secure.

   Correct The analogy that a chain is only as strong as its weakest link applies to secure programming. If the infrastructure is not secure, if the Libraries, the compiler, the link or loader, and so forth, the network services you rely on, aren't secure, or aren't robust, then it's very hard to build a program that's robust for a service that's robust on top of them.

10. Robust programming is important because:

- Doing it provides much-needed jobs for itinerant programmers
- Software engineering licensure requires understanding it
- Programs are ubiquitous, so a lot depends on them functioning correctly
- Without it, software firms would collapse

Correct Nowadays, computers are ubiquitous. They are everywhere; even in our bodies, controlling medical devices like pacemakers. If you have a bug or a security problem with a pacemaker, the person who has it is going to have a lot of trouble. For example, if I am able to tell the pacemaker, instead of 72 beats a minute, go to 300 beats a minute, or go to 0 beats a minute, that person is going to be either very sick or die.

11. What is the best way to ensure no data is added to or removed from the system with a DVD?

- Remove the DVD reader/writer from the computer.
- Add code to the kernel that turns the DVD reader/writer off, so it cannot be used.
- Add a filter that detects when a DVD is inserted, and blocks all input from, and output to, that DVD.
- Search all users to ensure they do not have a DVD.

Correct The easiest way to, incidentally, to make sure this policy is followed is simply to remove all DVD hardware from the system.

12. Why are assumptions made by a programmer and program so important to secure programming?

- The simpler the assumptions, the faster the program will run and the less chance of a vulnerability being found.
- The program should make no assumptions, as each assumption is tied to a programming error.
- The assumptions indicate which part of the program is most secure.
- The assumptions the program makes shows you what you have to trust.

Correct That points to the basic rule, if you know what your assumptions are, you're 95% of the way to a more secure system or program. Many of the assumptions are about what you trust. What you do is you write your program and ask what am I assuming here? What am I trusting? What happens if my assumption is wrong or my trust is misplaced?

13. A user does not trust the system administrator on a system. The administrator has root or ad min access. So the user writes a program to encrypt her files when not in use, and decrypt them when she need to use them. Assuming the cipher used cannot be broken easily (for example, RSA with 4096 bit keys, or AEC-192), how good is this procedure?

    - It's unnecessary, because all system administrators are trustworthy; the user is wrong in not trusting him.
    - It's adequate, because the user can get the files when she needs them, but she has to go through an extra step to use them (decryption) or store them (encryption).
    - It's very good, as the system administrator cannot read the encrypted files even though she can access any file on the system.
    - It's bad, because the system administrator can read the files as they are encrypted or decrypted.

    Correct Typically, you must assume that you can't secure anything from root. root can do anything it wants. The access that they can get using whatever you write is unlimited. Encryption doesn't help much here if the data is encrypted on the system because then root can always add a backdoor or a Trojan horse to the decryption program or the encryption program. The only way to make sure that data is not read is to put it on the system encrypted and pull it off the system encrypted.

14. When writing a secure, robust program, which of the following should you check or do? (Select all that apply.)

    - Error checking for non-cryptographic inputs in a cryptographic library

      Correct This is simply good programming practice to following in order to write a secure robust program.

    - Erasing any passwords at the end of the program
    - Check the form of results from a DNS query

      Correct This is simply input validation, the input being from the DNS server. It is a good practice to following in order to write a secure robust program.