Homework 2 - Software Security - CS3780

1.) Write two variations of a function (one that is reentrant and another that is non-reentrant) that calculates the sum of the numbers 1 to n (where integer n is a parameter of the function). The function can be in either C++ or Java.

Reentrant version (Java)

```java
public static int sumReentrant(int n) {
    int total = 0;        // local variable
    for (int i = 1; i <= n; i++) {
        total += i;
    }
    return total;
}
```

Non-reentrant version (Java)

```java
public class Calculator {
    private static int sharedTotal = 0;  // shared global
variable

    public static int sumNonReentrant(int n) {
        sharedTotal = 0;                 // resets shared
state
        for (int i = 1; i <= n; i++) {
            sharedTotal += i;
        }
        return sharedTotal;
    }
}
```

a) The reentrant version is safe because it uses only local variables, meaning nothing is shared between function calls. As a result, multiple threads can execute the function at the same time without interfering with one another. In contrast, the non-reentrant version is unsafe because it relies on a shared static variable (sharedTotal). If two threads call the function simultaneously, they will modify the same shared value, overwrite each other's data, and potentially cause race conditions.

2.) What are some signs of a possible race condition while testing an application?

- The program behaves differently every time you run it.
- Errors that are hard to reproduce or seem random.
- Data suddenly becomes incorrect or corrupted.
- Variables change unexpectedly.

- Program crashes only sometimes, not always.

What are some signs or behaviors that might indicate a race condition during manual or automated testing of a multithreaded application?

- Tests that fail intermittently (passes once, fails the next time).
- Logs that appear in a different order each run.
- Threads finishing in inconsistent sequences.
- Unexpected slowdowns or deadlocks.
- Values that should never happen (e.g., negative counters, missing items, duplicates).
- UI elements or output showing inconsistent results under load.

3.) Give an example in computer science where usability and security conflict that is not about passwords.

- A common example of a usability and security conflict that does not involve passwords is the use of CAPTCHAs. CAPTCHAs improve security by blocking automated bots, making it harder for attackers to abuse forms or create fake accounts. However, they also reduce usability because they slow users down, are often difficult to read, and can cause frustration. In many cases, the more secure the CAPTCHA becomes, such as by presenting more complex or distorted challenges, the less usable it is for real users.

4.) My recommendations for the tester:

- Run tests with more threads or higher system load to force timing issues.
- Add logging around shared variables or critical sections to catch unexpected behavior.
- Use thread sanitizers or concurrency testing tools (e.g., Java's -Xconcurrent, Thread Sanitizer, IntelliJ concurrency tools).
- Run the tests many times (loop the same test 1,000+ times).
- Use stress testing or randomized scheduling tools.
- Try different hardware speeds (slow down CPU or run on a VM).
- Force context switching by adding artificial delays (sleep() calls) during testing.

5.) Fingerprinting is a technique attackers use to gather unique characteristics about a system, software, or device in order to identify what it is running. By learning details such as the operating system, server version, installed software, and open ports, attackers can determine the system's potential weaknesses before launching an attack. For example, a web attacker may send specially crafted HTTP requests and analyze the responses to find out that a server is running Apache 2.4.41 on Ubuntu Linux. This information immediately reveals which known vulnerabilities apply to that specific version, allowing the attacker to plan a more targeted attack.

6.) Privacy and security are closely related but serve different purposes when it comes to protecting information. Privacy focuses on who is allowed to access your personal information and how that data is collected, shared, or used. Security, on the other hand, protects the confidentiality, integrity, and availability of data from attackers or unauthorized access. Neither one is always more important than the other, because their importance depends on the context. Security ensures that information is safe, while privacy determines how that protected information is handled. It is possible to have strong security without strong privacy, such as when data is encrypted but still heavily tracked, but strong privacy cannot exist without security because data must be protected before privacy can even be considered. In most systems, security forms the foundation, and privacy is built on top of it.