1.

1. The memory corruption CWE causes undefined behaviors, such as corruption or crashing the program.
2. The Input Authentication CWE could lead to SQL attacks if this code was implemented in a database, or DoS attacks.
3. The improper Neutralization of Input CWE can also lead to SQL attacks if implemented in a database.
4. The Improper Authorization CWE can lead to a user accessing admin privileges which is bad when they have malicious intentions.

2. I have written multiple programs without proper input validation, such as basic calculators. I don’t have many projects so this is as good a example as I have, but if I incorporated this in a database, it could have left the database susceptible to SQL injection attacks. To follow the Principle of Least Privilege and Minimal Trusted Computed Base, I need to make sure the program doesn’t have more privileges than it needs, and isolate security critical code and data from the rest of the system. For MTCB there are also other methods such as memory encryption too.

3. Say some online service lets users upload and execute their own scripts, they could potentially access unauthorized files, Cause DoS, leak data, and more. Language-based security could solve this problem by rewriting the scripts and removing dangerous sections of the code like forks and infinite loops. This is better than traditional security since it’s more flexible, performant, and secure.