Team Discussion:

What is a Secure Programming Language?

Question 1: What factors determine whether a programming language is secure or not?
The following primary activities are part of secure coding techniques or strategies:

Definition of areas of interest of the application:
Determine the application's critical and security-sensitive code and data assets.
Analysis of software architecture:
Review the programming for any noticeable security vulnerabilities. Secure communication between components to assist protect the integrity and confidentiality of data. Ensure if appropriate authentication and permission procedures are used to secure sensitive data. Make sure availability is incorporated across the entire architecture.
Review of implementation details:
Using secure coding methods, inspect the code. Ensure peer review is conducted with the purpose of identifying security gaps. Provide the developer comments and make sure the necessary adjustments are performed.
Verification of logic and syntax:
Verify the syntax and logic of the code to make sure there are no noticeable implementation flaws. Make sure the programming is carried out in accordance with the programming language/widely platform's accepted secure coding rules.
Whitebox/Unit Testing:
Along with testing to ensure functionality, the developer typically tests his code for security. Third-party data and/or APIs required for testing can be virtualized using mock data and/or APIs.
Blackbox Testing:
An expert QA engineer tests the programme for security flaws including unauthorised data access, paths that unintentionally expose code or data, weak passwords or hashes, etc. The testing results are communicated to all key stakeholders, including the architect, to ensure that the vulnerabilities found are mitigated.
Question 2: Could Python be classed as a secure language? Justify your answer
Python has a basic syntax, is generally readable, and often only has one explicitly stated way to do something. It includes a number of small, thoroughly tested standard library modules. The most current versions of Python 3.x provide fixes for a number of security flaws found in Python 2.x versions.
Evaluating arbitrary input: It should be noted that the same attack functions in Python 3 as well, but we must make some adjustments to the code object's parameters since in Python 3, code objects require a second argument. Additionally, certain parameters and the code-string must be of the byte type.
Overflow errors: Python reports an overflow error because len yields integer objects, and the real number is too huge to fit within an int. In Python 2, the function expecting an int object but receives long and causes a TypeError if the class is not derived from object. It depends on the application code and dependent module code's overflow error handling

and masking. Since Python is written in C, any overflow faults not properly handled in the underlying C code can lead to buffer overflow exceptions, where an attacker can write to the overflow buffer and hijack the underlying process, taking control of the programme. If a module or data structure can manage overflow errors and raise exceptions, code exploitation is less likely.

Serializing objects: Work-around to prevent such exploits: Use safe serialisation in your programmes and avoid using unsafe modules like pickle. Trust a more secure option like json or yaml instead. Use sand-boxing technologies or code jail to create safe environments that restrict the execution of malicious code on the system if your application truly depends on accessing the pickle module for whatever reason.

Question 3: Python would be a better language to create operating systems than C. Discuss.

Python is easier to create than C since it has fewer keywords and more flexible language syntax. When a variable is created in C, its type must be defined, and only values of that specific type are allowed to be given to it.