Secure Software Development

Team Discussion

What is a secure programming language?

1. **What factors determine a secure programming language?**

- Type safety.
- Type checking.
- Memory Management.
- Programming Paradigm.
- Nullability checks.
- Prevent buffer overflow.
- Prevents use of language constructs that can be misused.
- Eliminate vulnerabilities.
- Prevents injection errors.
- In-built functionality to prevent vulnerabilities.
- Has support to ensure confidentiality of data access.
- Has support to validate input of data.

2. **Is Python a secure programming language?**

Yes.

- Memory management.
- Easily readable, simple syntax. Code is less complicated.
- Follow best practices for secure code.
- Python helps to mitigate input and overflow errors.
- Used widely by the security industry.
- Has many well-tested libraries

No.

- String inputs are evaluated with no control or limits.
- Existing Python libraries has vulnerabilities.
- No truly secure mainstream language.

- No modern language is fully secure

3. **Is Python best suited to develop operating systems versus C?**

No.

- Operating systems require deeper security knowledge.
- Python is an interpreted language, and its performance will lag behind native C implementations.
- Python lacks true multithreading and not suited for CPU-intensive tasks.
- Python is a Very High-Level language and is not designed for low-level operations that access resources directly.
- Native host machine implementations are still implemented in C or assembly language.
- Python compiles code to bytecode which must be interpreted.
- Python does not have direct control over memory management, which is important in OS development.