Unit 3: Programming Languages: History, Concepts & Design

Team activity:
Discussion

Task:

You should read Chapter 2,6,7,8 of the course text (Pillai, 2017) and Cifuentes & Bierman (2019) and then answer the questions below, adding them as evidence to your e-portfolio.

1. What factors determine whether a programming language is secure or not?
2. Could Python be classed as a secure language? Justify your answer.
3. Python would be a better language to create operating systems than C. Discuss.

Answer:

1. Factors Determining Whether a Programming Language Is Secure

The security of a programming language depends on its ability to mitigate common vulnerabilities and provide robust safeguards. Key factors include:

- **Memory Safety:** Languages that automatically manage memory, such as Python and Java, prevent memory-related vulnerabilities like buffer overflows. Cifuentes and Bierman (2019) note that memory safety features significantly reduce risks associated with manual memory management. Conversely, languages that rely on manual memory allocation, such as C, are more prone to these issues.

- **Injection Prevention:** The ability of a language to prevent code injection attacks, such as SQL injection or cross-site scripting (XSS), is crucial. Secure languages offer built-in abstractions or frameworks (e.g., .NET's LINQ) to handle input securely (Cifuentes & Bierman, 2019). Input validation and type-checking further enhance protection against unintended code execution and data manipulation (Pillai, 2017).
- **Data Confidentiality:** Effective prevention of information leaks is essential for language security. While mainstream languages like Python lack built-in mechanisms for preventing data leaks, some experimental languages, such as Jeeves, incorporate faceted values for this purpose (Cifuentes & Bierman, 2019).
- **Community and Best Practices:** The strength of a language's community in promoting secure coding practices, providing patches, and updating libraries also influences security. Secure languages often discourage or restrict functions that enable arbitrary code execution (Pillai, 2017).

2. Could Python Be Classed as a Secure Language?

Python can be considered moderately secure due to several built-in features and safeguards, but it does not meet the criteria to be classified as highly secure.

- **Memory Safety:** Python's managed memory model ensures automatic memory allocation and garbage collection, minimizing the risks associated with memory leaks and buffer overflows (Cifuentes & Bierman, 2019; Pillai, 2017).
- **Injection Prevention:** While Python supports secure coding practices through libraries that encourage parameterized queries (e.g., sqlite3), it lacks inherent

mechanisms such as taint tracking, leaving room for human error and potential injection vulnerabilities (Cifuentes & Bierman, 2019).

- **Information Leak Prevention:** Python lacks built-in language-level features to address potential information leak errors. This can expose code to inadvertent data leakage through practices such as improper logging (Cifuentes & Bierman, 2019).

Despite these shortcomings, Python's straightforward syntax and large collection of libraries support secure programming practices when used carefully. Developers need to adhere to best practices, such as avoiding the use of eval with untrusted input and handling serialization with caution (Pillai, 2017). Therefore, Python's security largely depends on the discipline and knowledge of its developers.

3. Python Would Be a Better Language to Create Operating Systems Than C. Discuss.

The statement that Python would be a better language for creating operating systems than C is debatable due to the distinct nature of each language's capabilities.

- **Performance and Low-Level Control:** C is the preferred language for OS development due to its low-level access to system memory and hardware. This access is crucial for managing memory allocation manually and optimizing performance, ensuring stability and efficiency in the OS environment. Python, being an interpreted language with automatic memory management, introduces overhead that compromises performance (Cifuentes & Bierman, 2019; Pillai, 2017).
- **Security and Ease of Development:** Python offers higher memory safety and simpler, more readable code, which can reduce human error during

development. However, the trade-off is its lack of direct system-level control and the overhead of interpretation, which are significant drawbacks for OS development. These characteristics make Python more suited for rapid prototyping or high-level application development rather than core system functionalities (Pillai, 2017).

- **Language Suitability for OS Development:** While Python could be used for scripting OS components or as an auxiliary tool in development, it lacks the execution speed and system-level efficiency provided by C. C's use of pointers and its compatibility with hardware make it irreplaceable for building robust, performance-driven operating systems (Cifuentes & Bierman, 2019).

In conclusion, while Python promotes safer and faster development cycles, it does not offer the low-level access or performance efficiency required for developing full operating systems. C remains the superior choice for this purpose due to its unparalleled control over system resources and execution speed.

References

Pillai, A.B. (2017) *Software architecture with python: Design and architect highly scalable, robust, clean, and high performance applications in Python*. Birmingham, UK: Packt Publishing.

Cifuentes, C.G., & Bierman, G.M. (2019). What is a Secure Programming Language? Summit on Advances in Programming Languages.