Team Discussion: What is a Secure Programming Language?

You should read Chapters 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?**

Secure coding is creating software from the ground up with security in mind. This entails early security requirement identification, threat modelling to foresee potential risks, and the application of secure coding techniques. As stated by Pillai (2017), significant duties include identifying critical assets, assessing software design, analysing implementation details, confirming logic and syntax, doing unit and Blackbox testing, and addressing any security flaws found.

According to Cifuentes & Bierman (2019), there are three principal vulnerabilities:

1.1 **Buffer Errors:** Buffer overflow attacks overload memory storage and can corrupt memory beyond the buffer's boundaries. The attack can be stack-based or heap-based.
1.2 **Injection Errors:** Injection errors have several vulnerabilities, including XSS, SQL injection, code injection, and OS command injection. XSS allows attackers to inject malicious code into websites, compromising data confidentiality and integrity. Websites are vulnerable if they display unsanitised user-supplied data. There are three types of XSS: reflected, stored, and DOM-based.
1.3 **Information Leak Errors:** Information leakage releases information to an untrusted environment. It can occur through various means, including log files, caching, and error messages. The most common type is through log files.

The wrong abstractions are used in programming languages, which results in vulnerable code. Information leaks come from manually tracking sensitive data, whereas buffer and injection mistakes are caused by manual pointer and string management, respectively.

Any secure programming language deserving of the designation must have first-class support for each of the three categories, according to Cifuentes & Bierman's (2019) thesis. Consequently, a secure programming language must provide best-in-class linguistic support, including memory security, data integrity, and secrecy methods to solve these issues.

Developers need safe abstractions to write secure code rather than migrating millions of lines of insecure code.

2. **Could Python be classed as a secure language? Justify your answer.**

Python's syntax is simple, readable, and has a clear way of doing things. Moreover, it boasts a well-tested set of standard library modules, indicating that it is secure (Pillai, 2017).

However, Pillai (2017) states that security concerns related to Python encompass various aspects, such as evaluating expressions, reading input, overflow errors, and serialisation issues. Web applications created using frameworks like Django and Flask are particularly vulnerable. To mitigate these risks, it is imperative to employ secure coding practices such as avoiding pickle exec and ensuring integer overflow protection. Additionally, it is crucial to adopt safer input methods.

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

An operating system (OS) manages application programs and provides services to users through a command-line interface (CLI) or graphical user interface (GUI) (Bigelow, 2021).

It needs a solid foundation in computer science fundamentals, basic programming, and high-level and low-level programming languages to develop an operating system. Assembly languages directly communicate with the CPU, while x86 architecture and C programming are commonly used for OS development (saijyosthanakanchi555, 2021).

Dougvj (2012) explains that Python is a high-level programming language that requires an abstraction layer, like the Kernel, to access hardware and perform low-level data structure manipulation. It can create an operating system with only the low-level components written in C and assembly and the majority written in Python. On the one hand, writing an OS in assembly code is feasible but time-consuming and skill-intensive. OS development is made simpler by high-level languages like C and C++, with C being beneficial as a middle-level language with both high-level constructs and low-level features. C was created with system-level and embedded program development in mind. It is a highly portable structured language that enables the division of extensive programs into more straightforward functions, making it suitable for scripting system applications on Windows, UNIX, and Linux (Dahanayaka, 2021).

Operating systems are typically written in C, a portable and powerful programming language that can create complex designs. While assembly language is used for some parts of the Windows kernel, C is used for the seed as a whole. Other programming languages like Python are also options, but C remains the best choice for system programming due to its low and high-level operations (Lemp.io, 2023).

Formative activities: Collaborative discussion in Unit 1 – Summary Post

My previous post discussed Cross-Site Scripting (XSS) and its potential security risks. XSS is a type of vulnerability that occurs when malicious scripts are inserted into a trusted website or app, enabling hackers to steal data, take control of accounts, and execute harmful code on the user's browser (OWASP, N.D.). This issue arises when applications fail to validate, filter, or sanitise user-supplied data. To illustrate the weaknesses of XSS, I created a Sequence Diagram. However, my tutor suggested using an Activity Diagram, which visually represents a system's actions or control flow more effectively (Smartdraw, N.D.). I agree and have included an alternative activity diagram to demonstrate a malicious XSS attack based on the ideas of Gupta & Sharma (2012).

During a recent discussion, my colleague Sebastian raised an essential point regarding XSS vulnerabilities. Selvamani et al. (2010) found that careless use of the Document Object Model in JavaScript can create vulnerabilities where malicious code from another page can alter the principle of the first page on a local system. It is essential to understand these distinctions, as each one requires specific countermeasures. Sebastian ended his point with a relevant question: How can developers protect their applications against various XSS attacks? IBM (N.D.) suggests that to prevent XSS attacks; the application must validate all input data, only allow listed data, and encode all variable output on pages before returning it to the user.

I have commented on my colleagues' posts regarding cryptographic flaws and insecure design and implementation of APIs.

After analysing colleague Adesola's post on Cryptographic Flaws, I realised the importance of web application security. Further research revealed potential

implications such as data breaches, loss of trust, legal and compliance issues, and intellectual property theft, as explained in Ali's research (2023).

My colleague Sebastien provided insightful information that helped me understand the importance of security in API projects. I learned more about RESTful, BOLA, and UBER API security incidents, illustrating the need for in-API security.

References:

Pillai, A.B. (2017). *Software architecture with Python*. Packt Publishing Ltd.

Cifuentes, C. & Bierman, G. (2019). What is a secure programming language?. In *3rd Summit on Advances in Programming Languages (SNAPL 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Bigelow, S. (2021). *What is an Operating System (OS)? Definition, Types and Examples*. [online] WhatIs.com. Available at: https://www.techtarget.com/whatis/definition/operating-system-OS.

saijyosthanakanchi555 (2021). *Guide to Build an Operating System From Scratch*. [online] Available at: https://www.geeksforgeeks.org/guide-to-build-an-operating-system-from-scratch/.

Dougvj (2012). *Is it possible to create an operating system using Python?* [online] Available at: https://stackoverflow.com/questions/10904721/is-it-possible-to-create-an-operating-system-using-python [Accessed 26 Aug. 2023].

Dahanayaka, M. (2021). *Create Your Own Operating System*. [online] Medium. Available at: https://medium.com/@mekaladahanayaka80/create-your-own-operating-system-a4b1c179c28f#:~:text=Using%20a%20high%2Dlevel%20language [Accessed 27 Aug. 2023].

Lemp.io. (2023). *HOW TO BUILD AN OPERATING SYSTEM IN C.* Available at: https://lemp.io/how-to-build-operating-system-in-c/ [Accessed 27 Aug. 2023].

OWASP (N.D.). *Cross-Site Scripting (XSS) Software Attack | OWASP Foundation*. [online] Available at: https://owasp.org/www-community/attacks/xss/#:~:text=Cross%2DSite%20Scripting%20(XSS).

Smartdraw (N.D.). *Activity Diagram - Activity Diagram Symbols, Examples, and More*. [online] Available at: https://www.smartdraw.com/activity-diagram/#:~:text=Learn%20More-.

Gupta, S. & Sharma, L. (2012). *Exploitation of Cross-Site Scripting (XSS) Vulnerability on Real World Web Applications and its Defense.* Available at: https://research.ijcaonline.org/volume60/number14/pxc3883594.pdf

Selvamani, K., Duraisamy, A. & Kannan, A. (2010). *Protection of Web Applications from Cross-Site Scripting Attacks in Browser Side*. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.1004.1769.

IBM (N.D.). *Protect from cross-site scripting attacks*. Available at: https://www.ibm.com/garage/method/practices/code/protect-from-cross-site-scripting/.

Ali, Z. (2023). *Cryptographic Failures: Understanding the Pitfalls and Impact.* [online] Available at: https://www.linkedin.com/pulse/cryptographic-failures-understanding-pitfalls-impact-zahid-ali/.