# 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?
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.

1. Cifuentes & Bierman (2019: 6) stated that a programming language is secure if they have no buffer and injectors errors and no information leaks.

Buffer errors are related to memory issues. An example is the buffer overflow error, where memory storage can get overloaded. An attacker uses this method to overwrite and damage the memory outside the buffer limits. Depending on the concrete technical design of the buffer overflow, there are different types, such as stack overflow, heap overflow, integer overflow, pointer overflow or string overflow. Suppose data can be written to a memory area that is not intended for this purpose. In that case, various consequences can occur, such as programme crashes, compromise of data, obtaining extended rights or execution of malicious code (Cifuentes & Bierman, 2019: 3).

Injectors errors are, for example, cross-site scripting (XSS) and SQL injection. Cross-site scripting enables attackers to apply malicious code to a website which can destroy the application. SQL injection allows attackers to execute any SQL code. The most typical example is the SQL injection, where an attacker sends "101 OR 1=1" instead of just "101". When this data is inserted into an SQL query, the meaning changes so that ALL records are returned instead of just one. Anything with a "command interface" that combines data into command is vulnerable. Even XSS is just a form of HTML injection (Cifuentes & Bierman, 2019: 3-4).

Information leaks are sensitive data leaked through log files, caching, environment variables, test code, shell error messages, servlet runtime error messages, Java runtime error messages, etc. The most common information leaks come from log files where users inadvertently provide their location when failing to log in to an application. The user's location will occur in the log file (Cifuentes & Bierman, 2019: 5).

2. Cifuentes & Bierman (2019: 6) claims that none of the programming languages which are currently used is secure. Nevertheless, Pillai (2017: 290) states that Python is very readable and understandable through its simple syntax. Furthermore, it includes well-tested libraries. However, ultimately, every language has weaknesses, also Python. Security vulnerabilities like buffer overflow, input validation issues, access control issues, cryptographic weaknesses, information leaks, and insecure file operations also occur with Python (Pillai, 2017: 326).

An example is a buffer overflow error with Python:

```
# len_overflow.py

class A(object):
def __len__(self):
return 100 ** 100
class B:
```

```
def __len__(self):
return 100 ** 100
try:
len(A())
print("OK: 'class A(object)' with 'return 100 ** 100' – len calculated")
except Exception as e:
print("Not OK: 'class A(object)' with 'return 100 ** 100' – len raise Error: " + repr(e))

try:
len(B())
print("OK: 'class B' with 'return 100 ** 100' - len calculated")
except Exception as e:
print("Not OK: 'class B' with 'return 100 ** 100' - len raise Error: " + repr(e))
```

The output of the code above:

```
$ python3 len_overflow.py
Not OK: 'class A(object)' with 'return 100 ** 100' - len raise Error:
OverflowError("cannot fit 'int' into an index-sized integer",)
Not OK: 'class B' with 'return 100 ** 100' - len raise Error:
OverflowError("cannot fit 'int' into an index-sized integer",)
```
(Pillai, 2017: 299-300).

The problem is that the len method returns an int too large to fit into an int. Therefore, Python performs an overflow error. It depends on which method and application code are used and how they can handle the overflow problem. However, since the buffer overflow vulnerability exists, attackers can still crash the program. With the right exceptions through modules or data structure, code exploitation can be mitigated if an overflow error occurs and code execution is stopped (Pillai, 2017: 300).

3. Python is a high-level interpreted language. Because of its syntax, it is easy to learn and an excellent programming language for people who would like to jump into the programming world. It was released in 1991, and nowadays, it contains many libraries for developing anything. Because of its reusability through the object-oriented programming structure, standard libraries can be used again, and the development of an application is

faster than in other programming languages like C. However, it is not used for developing an operating system. Since it is an interpreted language, it has speed limitations compared to a compiled language like C. In addition, through its garbage collector capability, the storage space cannot be managed manually. Since an operating system cannot know what the memory contains and what is garbage. On the contrary, the compiled programming language C was created for constructing a Unix system. Because of its high-speed performance, compactness, portability and flexibility, it is used for creating an operating system. Furthermore, it does not have a garbage collector; it manages the memory manually. In addition, the operating system Linux is developed with C; hence it also shows longevity. Last but not least, C is a middle-level language that combines the features of both high- and low-level languages. Low-level languages provide no abstraction from the commands of a computer. The structure of the commands or functions of the low-level languages resembles the instructions of a processor and is, therefore, suitable for creating an operating system (InterviewBit, 2022; OpenEDG, 2021).

**References:**

Pillai, A.B. (2017) *Software Architecture with Python*. Birmingham, UK. Packt Publishing Ltd.

Cifuentes, C. & Bierman, G. (2019) *What is a Secure Programming Language?* 3rd Summit on Advances in Programming Languages (SNAPL).136(3): 1 - 15.

InterviewBit (2022) Difference Between C and Python. Available from: https://www.interviewbit.com/blog/difference-between-c-and-python/ [Accessed 8 October 2022].

OpenEDG (2021) Learn C and C++, the languages of the past and today. Available from: https://openedg.org/learn-c-and-c-the-languages-of-the-past-and-today/ [Accessed 8 October 2022].