Codio Activity - Exploring Python tools and features

Part I

In this example, you will compile and run a program in C using the **Codio workspace** provided (Buffer Overflow in C). The program is already provided as bufoverflow.c - a simple program that creates a buffer and then asks you for a name, and prints it back out to the screen.

This is the code in bufoverflow.c (also available in the Codio workspace):

```
#include <stdio.h>
int main(int argc, char **argv)
{
char buf[8]; // buffer for eight characters
printf("enter name:");
gets(buf); // read from stdio (sensitive function!)
printf("%s\n", buf); // print out data stored in buf
return 0; // 0 as return value
{
```

Now compile and run the code. To test it, enter your first name (or at least the first 8 characters of it) you should get the output which is just your name repeated back to you.

Run the code a second time (from the command window this can be achieved by entering. /bufoverflow on the command line). This time, enter a string of 10 or more characters.

- What happens?

The compiler generated the **stack smashing detected error** in response to its defense mechanism against buffer overflows.

- What does the output message mean?

Buffer overflow is a security vulnerability that occurs frequently and is often exploited for attacks. A buffer overflow vulnerability can affect both locally operating software, internet services, and web applications. A buffer overflow occurs when it is possible to write more data into a reserved memory area (buffer) than the buffer is designed for. As a result, neighbouring memory areas are filled with data. 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, compromising data, obtaining extended rights or executing malicious code. Programming errors or conceptual weaknesses in software often cause vulnerability to a buffer overflow.

The causes for a buffer overflow are manifold. Often the vulnerability can already be found in the architecture of the computer systems, for example, if data and programmes

are located in the same memory. Programming languages are also a cause of vulnerability to buffer overflows. Some programming languages, such as **C or C++**, offer only limited possibilities to monitor compliance with the limits of memory areas automatically. Other causes include insufficient checking of user input or data transfers via data interfaces (Luber & Schmitz, 2021).

Part II

Now carry out a comparison of this code with one in Python (Buffer Overflow in Python), following these instructions:

In the Codio workspace, you will be using the file called Overflow.py:

```
buffer=[None]*10
for i in range (0,11):
    buffer[i]=7
print(buffer)
```

- Run your code using: Python overflow.py (or use the codio rocket icon)
- What is the result?

Output: IndexError: list assignment index out of range

- Read about Pylint at http://pylint.pycqa.org/en/latest/tutorial.html
- Install pylint using the following commands:

  pip install pylint (in the command shell/ interpreter)

- Run pylint on one of your files and evaluate the output:

  pylint your_file

- (Make sure you are in the directory where your file is located before running Pylint)
- What is the result? Does this tell you how to fix the error above?

Output:

Module Overflow

Overflow.py:4:0: C0303: Trailing whitespace (trailing-whitespace)

Overflow.py:5:0: C0304: Final newline missing (missing-final-newline)

Overflow.py: 1:0: C0103: Module name "Overflow" doesn't conform to snake_ca

(se naming style (invalid-name)

Overflow.py: 1:0: C0114: Missing module docstring (missing-module-docstring)

→ The Pylint testing doesn’t tell how to fix the index error (Buff overflow error)

Info to Python:

Use of programming languages that are less prone to buffer overflows and control the adherence to allocated memory areas, such as Java, Python or C# (Luber & Schmitz, 2021)

Be prepared to discuss your thoughts on both exercises at next week’s seminar.

Remember to record this into your e-portfolio.

References:

Luber, S. & Schmitz, P (2021) Was ist ein Pufferüberlauf (Buffer Overflow)?. Available from: https://www.security-insider.de/was-ist-ein-pufferueberlauf-buffer-overflow-a-1052439/ [Accessed 6 October 2022].