

Codio Activity

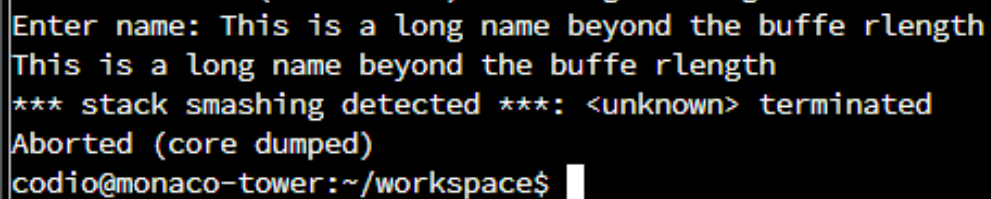
Exploring Python Tools and Features

Part 1: Buffer overflow

```
#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
}
```

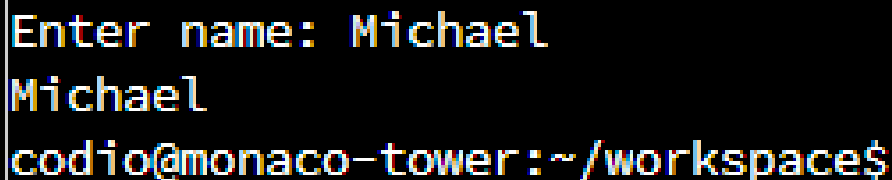
This code was executed twice. For the first execution, I input a text string such as “This is a long name beyond the buffer length” and received the following output:

A terminal window with a black background and yellow text. The prompt is 'codio@monaco-tower:~/workspace\$'. The user has entered 'This is a long name beyond the buffer length'. The program output is 'Enter name: This is a long name beyond the buffer length' followed by 'This is a long name beyond the buffer length'. Then, it displays a red error message: '*** stack smashing detected ***: <unknown> terminated' and 'Aborted (core dumped)'.

```
Enter name: This is a long name beyond the buffer length
This is a long name beyond the buffer length
*** stack smashing detected ***: <unknown> terminated
Aborted (core dumped)
codio@monaco-tower:~/workspace$
```

Such an output of “Stack smashing detected” indicates that a buffer overflow was detected by the gcc. The “smashing” refers to the stack being completely override with data. Behind the scenes, the gcc added protection variables (known as “stack canaries”) (Binary Exploitation, ND; RIT 2017) used to prevent buffer overflows from occurring.

For the second execution, I input text that correctly fits within the buffer. As expected, the program correctly reflected the input text back to the console.

A terminal window with a black background and yellow text. The prompt is 'codio@monaco-tower:~/workspace\$'. The user has entered 'Michael'. The program output is 'Enter name: Michael' followed by 'Michael'.

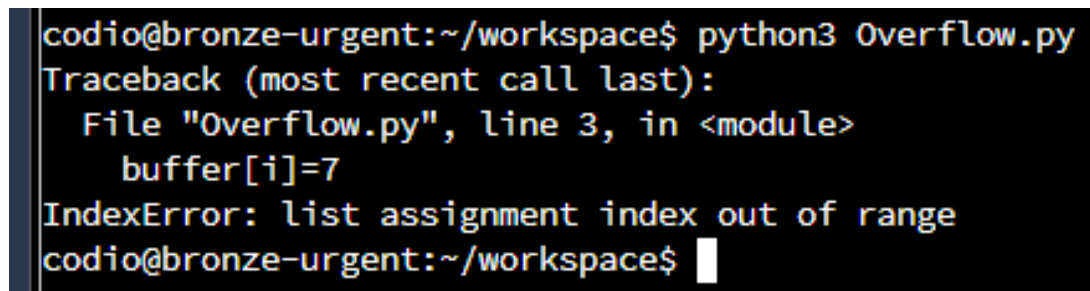
```
Enter name: Michael
Michael
codio@monaco-tower:~/workspace$
```

Part 2: Using linters

For this part, the same buffer overflow activity was carried out in Python.

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

The result of the first execution produced this error:

A terminal window with a black background and yellow text. The prompt is 'codio@bronze-urgent:~/workspace\$'. The command 'python3 Overflow.py' has been executed. The output shows a 'Traceback (most recent call last):' followed by 'File "Overflow.py", line 3, in <module>' and 'buffer[i]=7'. The error message is 'IndexError: list assignment index out of range'. The prompt 'codio@bronze-urgent:~/workspace\$' is shown again with a cursor.

```
codio@bronze-urgent:~/workspace$ python3 Overflow.py
Traceback (most recent call last):
  File "Overflow.py", line 3, in <module>
    buffer[i]=7
IndexError: list assignment index out of range
codio@bronze-urgent:~/workspace$
```

Here, Python code is attempting to set a value at the 11th index value based on the code `for i in range (0,14):`. Since the original buffer is initialised to ten elements (`buffer=[None]*10`), any assignment to the buffer *beyond* the allocated size, results in the `IndexError`.

Next, I installed **pylint** tool and ran analysis over the `Overflow.py` source code:

```

No config file found, using default configuration
***** Module overflow.py
F: 1, 0: No module named overflow.py (fatal)
codio@bronze-urgent:~/workspace$ pylint Overflow.py
No config file found, using default configuration
***** Module Overflow
C: 1, 0: Exactly one space required around assignment
buffer=[None]*10
    ^ (bad-whitespace)
C: 2, 0: Exactly one space required after comma
for i in range(0,14):
    ^ (bad-whitespace)
C: 3, 0: Exactly one space required around assignment
    buffer[i]=7
        ^ (bad-whitespace)
C: 4, 0: Trailing whitespace (trailing-whitespace)
C: 5, 0: Final newline missing (missing-final-newline)
C: 5, 0: Unnecessary parens after 'print' keyword (superfluous-parens)
W: 1, 0: Redefining built-in 'buffer' (redefined-builtin)
C: 1, 0: Module name "Overflow" doesn't conform to snake_case naming style (invalid-name)
C: 1, 0: Missing module docstring (missing-docstring)
C: 1, 0: Constant name "buffer" doesn't conform to UPPER_CASE naming style (invalid-name)

-----
Your code has been rated at -15.00/10

```

Pylint provides a number of *syntax formatting* improvements, but none of the recommendations identify the potential `IndexError` that can be raised by `buffer[i]=7`.

References

Binary Exploitation (ND) Stack Canaries. Available from

<https://ir0nstone.gitbook.io/notes/types/stack/canaries> [Accessed 02 Sep. 2021]

RIT (2017) RITCSEC Computing Security Blog: Buffer overflows, ASLR and Stack

Canaries. Available from <https://ritcsec.wordpress.com/2017/05/18/buffer-overflows-aslr-and-stack-canaries/> [Accessed 02 Sep. 2021]