

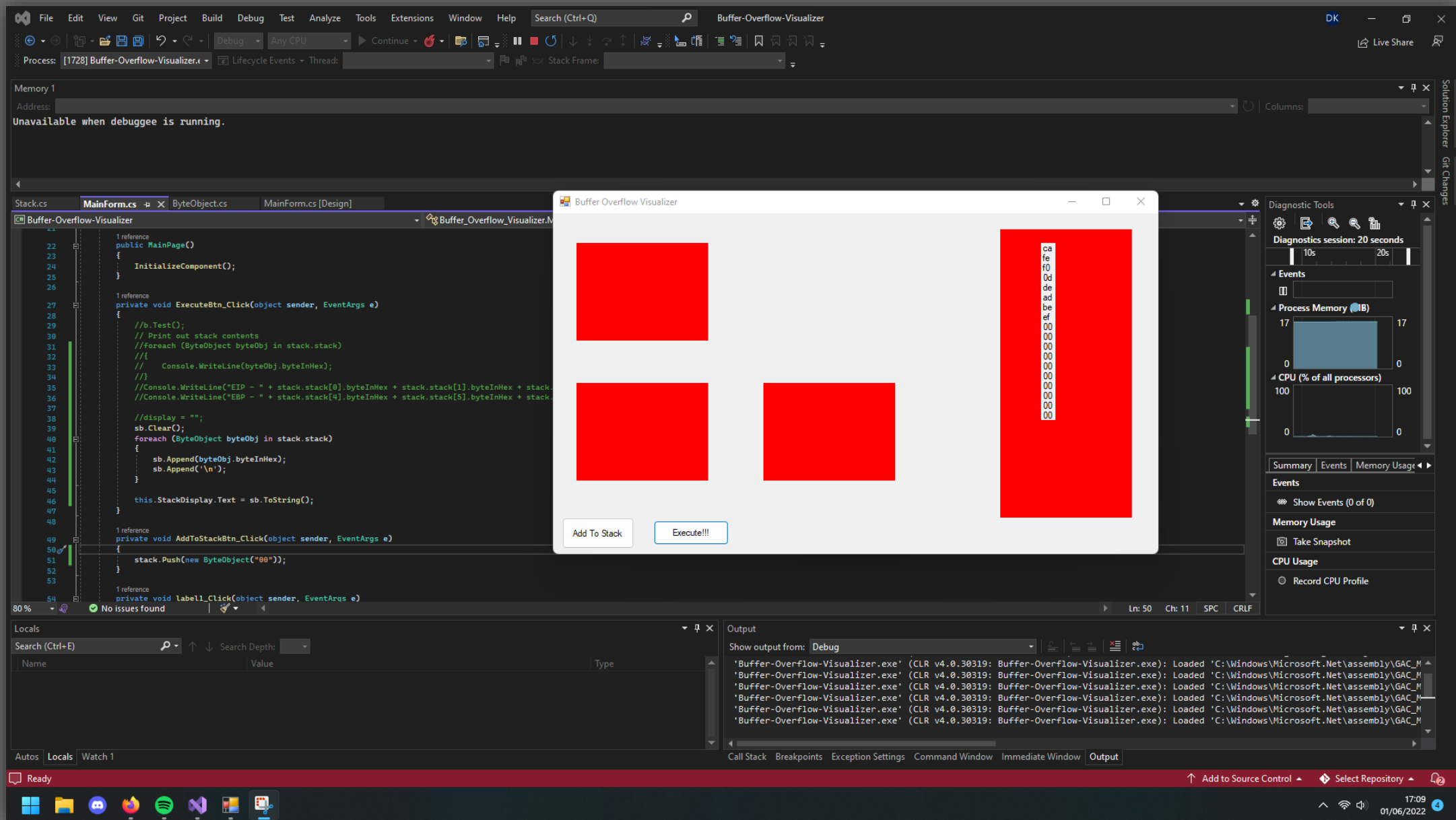
Buffer Overflow Visualizer

Python Tkinter Project

Devlog



1st June 2022 (01/06/2022)



2nd June 2022 (02/06/2022)

The image shows a Visual Studio IDE with a buffer overflow visualization tool. The tool has a window titled "Buffer Overflow Visualizer" with a red background. It features a stack diagram with a blue arrow pointing to the top of the stack. Below the stack, there is an "Input:" field with the value "A". To the right of the stack, there is a vertical list of memory addresses and their corresponding values: ca, fe, f0, 0d, de, ad, be, ef, 00, 00. At the bottom of the window, there are two buttons: "Add To Stack" and "Execute!!!".

The background shows the Visual Studio IDE with the following components:

- Code Editor:** Displays C# code for a buffer overflow demonstration. The code includes a `Console.WriteLine` statement, a `string val` declaration, a `Console.ReadLine()` call, and a `for` loop that iterates over the input string and pushes each character onto a stack.
- Solution Explorer:** Shows the project structure with files like `MainForm.cs`, `Stack.cs`, `ByteObject.cs`, and `Program.cs`.
- Output Window:** Displays the output of the program, showing the input string "A" and the stack contents.
- Diagnostic Tools:** Includes a "Diagnostics session: 17 seconds" section with a timeline and a "Process Memory" graph.
- Locals Window:** Shows the current state of local variables, including `bufSize`, `val`, and the stack.

```
52 Console.WriteLine("What do you wish to enter?: ");
53 string val;
54 val = Console.ReadLine();
55 int len_of_value = val.Length;
56
57 for (int i = 0; i < len_of_value; i++)
58 {
59     int index = (stack.Count() - 1) - i;
60     if (index == -1) break;
61     Console.WriteLine("Overriding " + stack[index].byteInHex + " with " + val[i] + " at");
62     ByteObject b = new ByteObject(val[i].ToString());
63     stack[index] = b;
64 }
65
66 // Print out stack contents
67 foreach (ByteObject byteObj in stack)
68 {
69     Console.WriteLine(byteObj.byteInHex);
70 }
71
```

Output:

```
bufSize = 43
bufSize = 44
val = A
Added 00 to stack
Added 00 to stack
Displaying stack contents...
```

4th June 2022 (04/06/2022)

File Edit View Git Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q)

Process: [2612] Buffer-Overflow-Visualizer.v... Lifecycle Events - Thread... Stack Frame...

Memory 1
Address:
Unavailable when debuggee is running.

MainForm.cs [Design] Stack.cs Main...
Buffer-Overflow-Visualizer
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
1 reference
private void InputSize_S...
{
// Set input length
inputLength = InputS...
charsLeft = InputSize...
InputLabel.Text = "I...
Console.WriteLine("i...
1 reference
private void BufSlider_S...
{
// Set buf size
bufSize = BufSlider.V...
BufSizeLabel.Text = ...
Console.WriteLine("bufSize = " + bufSize);
CodeViewLabel.Text = "# include <stdio.h>\n\n";
CodeViewLabel.Text += "int main()\n\n";
CodeViewLabel.Text += "{\n\n";
CodeViewLabel.Text += "int bufSize = 1;\n\n";
CodeViewLabel.Text += "printf(\"Hello :D);\n\n";
CodeViewLabel.Text += "printf(\"Enter value :);\n\n";
CodeViewLabel.Text += "char buf[bufSize];\n\n";
CodeViewLabel.Text += "gets(buf);\n\n";
CodeViewLabel.Text += "printf(\"Goodbye :));\n\n";
CodeViewLabel.Text += "};\n\n";
145
146

80 %
No issues found

Buffer Overflow Visualizer

#include <stdio.h>
int main()
{
int bufSize = 1;
printf("Hello :D);
printf("Enter value :);
char buf[bufSize];
gets(buf);
printf("Goodbye :);
}

Buf Size: 3
Input (0 characters left)
ABCDE

Registers:
EIP - cafe00d
EBP - deadED

Execute!!!

Reset

cafe00deadEDCBA

Diagnostic Tools
Diagnostics session: 28 seconds
Events
Process Memory (18)
CPU (% of all processors)

Summary Events Memory Usage
Events
Show Events (0 of 0)
Memory Usage
Take Snapshot
CPU Usage
Record CPU Profile

Locals
Search (Ctrl+E)
Name Value Type

Output
Show output from: Debug
Overriding 00 with A at index 10
Overriding 00 with B at index 9
Overriding 00 with C at index 8
Overriding ef with D at index 7
Overriding be with E at index 6
Displaying stack contents...

Call Stack Breakpoints Exception Settings Command Window Immediate Window Output

Ready
Add to Source Control Select Repository
11:45
04/06/2022

14th June 2022 (14/06/2022)

File Edit Selection View Go Run Terminal Help

buffer_overflow_visualizer.py - Buffer Overflow Visualizer - Visual Studio Code

EXPLORER

buffer_overflow_visualizer.py
1 from tkinter import
2 from buffer import Buffer

Code View

1

Program View

Register View

Execute

Reset

Info Box

Stack View

Buffer Overflow Visualizer

#include <stdio.h>
int main()
{
int bufSize = 5;
printf("Hello :D");
printf("Enter value:");
char buf[bufSize];
gets(buf);
printf("Goodbye :D");
}

user@machine: ~/home\$./program
Hello :D
Enter Value: AAAAAWOEM
Segmentation fault (core dumped)
user@machine: ~/home\$

EIP - cafef00d
EBP - MEOW

Buf Size: 5

Input (0 chars left)
AAAAAWOEM

EIP, ca
EIP, fe
EIP, f0
EIP, 0d
EBP, M
EBP, E
EBP, O
EBP, W
Buf[4], A
Buf[3], A
Buf[2], A
Buf[1], A
Buf[0], A

Execute!!!

Reset

Pushed 41 onto stack
Overriding 41 with A at index 39
Created a byte of value A
Overriding 41 with A at index 38
Created a byte of value A
Overriding 41 with A at index 37
Created a byte of value A
Overriding 41 with A at index 36
Created a byte of value A

OUTLINE
TIMELINE

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.10.4 64-bit

11:26
14/06/2022

14th June 2022 (14/06/2022)

The screenshot displays a Buffer Overflow Visualizer window overlaid on a Visual Studio Code editor. The visualizer window is titled "Buffer Overflow Visualizer" and contains several interactive elements:

- Code View:** A button to view the source code of the program.
- Program View:** A button to view the current state of the program, including the stack and memory.
- Execute:** A button to execute the program.
- Reset:** A button to reset the program state.
- Info Box:** A button to view information about the program.

The visualizer shows a memory buffer with hexadecimal values. The stack pointer (ESP) is currently at 0x00000000, and the program counter (PC) is at 0x00000000. The buffer contains the following values (from top to bottom):

- Buf[42]: 00 0x7fb583c
- Buf[44]: 00 0x7fb5838
- Buf[43]: 00 0x7fb5834
- Buf[42]: 00 0x7fb5830
- Buf[41]: 00 0x7fb582c
- Buf[40]: 00 0x7fb5828
- Buf[39]: 00 0x7fb5824
- Buf[38]: 00 0x7fb5820
- Buf[37]: 00 0x7fb581c
- Buf[36]: 00 0x7fb5818
- Buf[35]: 00 0x7fb5814
- Buf[34]: 00 0x7fb5810
- Buf[33]: 00 0x7fb580c
- Buf[32]: 00 0x7fb5808
- Buf[31]: 00 0x7fb5804
- Buf[30]: 00 0x7fb5800
- Buf[29]: 00 0x7fb57fc
- Buf[28]: 00 0x7fb57f8
- Buf[27]: 00 0x7fb57f4
- Buf[26]: 00 0x7fb57f0
- Buf[25]: 00 0x7fb57ec
- Buf[24]: 00 0x7fb57e8
- Buf[23]: 00 0x7fb57e4
- Buf[22]: 00 0x7fb57e0
- Buf[21]: 00 0x7fb57dc
- Buf[20]: 00 0x7fb57d8
- Buf[19]: 00 0x7fb57d4
- Buf[18]: 00 0x7fb57d0
- Buf[17]: 00 0x7fb57cc
- Buf[16]: 00 0x7fb57c8
- Buf[15]: 0x7fb57c4
- Buf[14]: 0x7fb57c0
- Buf[13]: 0x7fb57bc
- Buf[12]: 0x7fb57b8

The background shows the Visual Studio Code editor with a Python file named "buffer.py" and a terminal window showing the output of the program. The terminal output indicates that the program is overriding memory at index 45, 44, 43, and 42 with the value 'A'.

24th June 2022 (24/06/2022)

Buffer Overflow Visualizer

Endianness Architecture Screen Resolution

```
#include <stdio.h>

int main()
{
    int bufSize = 6;

    printf("Hello :D");
    printf("Enter value: ");

    char buf[bufSize];
    gets(buf);

    printf("Goodbye :)");
}
```

Danioooo@machine: ~/home\$./program
Hello :D
Enter Value: AAAAAAAA
Segmentation fault (core dumped)
Danioooo@machine: ~/home\$

Object	Value	Memory
EIP	cafef00d	0xf7fb5870
EBP	dead4141	0xf7fb586c
Buf[5]	41	0xf7fb5868
Buf[4]	41	0xf7fb5864
Buf[3]	41	0xf7fb5860
Buf[2]	41	0xf7fb585c
Buf[1]	41	0xf7fb5858
Buf[0]	41	0xf7fb5854

Execute

AAAAAAA

Buf Size:
6

EIP = cafef00d
EBP = dead4141

Reset