

Lab: Stack and Stack Frame

Mayur Suresh

CSC 472-02: Software Security 2

Date: October 2nd, 2022

In this lab we will use MobaXterm to go through the lab1 C program which has 2 functions, the main function and the add_plus1() function. We use Badger CTF and use gdb to disassemble the lab1 program to see what is inside a program with assembly code. We will also see how memory is handled and moved according to the program with eight 32-bit registers, the stack, and the stack frame. We can see how each line of the program works but setting a break line with the `br*` memory address and then running (`r`) in order to run the following line the `si` command is used to see how each line interacts behind the scene.

Assembly code that creates the stack frame followed by their memory addresses are:

Source Image 1

0x08049195: PUSH EBP

0x08049196: MOVE EBP, ESP

The meaning of the lines:

0x080491b1: mov DWORD PTR [ebp-0x10],0x5

0x080491b8: mov DWORD PTR [ebp-0xc],0x6

The first line means moving the 32-bit representation of 0x5 into 4 bytes at address [ebp-0x10]

The second line means moving the 32-bit representation of 0x6 into 4 bytes at address [ebp-0xc]. So, this means that int x is 5 and int y is 6 as shown in the lab1.c.

The reason why on the stack that the values of 6 and 5 are listed twice on the stack is because the initial 6 and 5 at the addresses (according to Source Image 2) 0xffffd6b8 and 0xffffd6bc are the storing of the 6 and 5 to the int a and int b addresses in the main function. While the other 5 and 6 at the addresses (according to Source Image 2) 0xffffd6a8 and 0xffffd6ac are there when the add_plus1 function is called in the main function and then passing through the a and b variables to this function.

Upon disassembling the add_plus1() function the first 2 assembly code are the code that are implemented in order to start the stack frame for the add_plus1() function.

After going through line by line as shown in Source Image 3 that after the ADD EAX, EDX it is inferred that the result is stored at the Register EAX as also shown on the stack where the EAX register has the summation.

All in all, this lab's purpose was to show what happens behind the scene of a program. How when data stored in a variable is moved to various registers to then push it to the method when called. This lab also showed us how the stack and the stack frame behave when data is moves around. It also showed how when a function is called a new stack frame is implemented.

Source Images Below.

Source Images

```
gef> disas main
Dump of assembler code for function main:
0x0804918b <+0>: lea ecx,[esp+0x4]
0x0804918f <+4>: and esp,0xffffffff
0x08049192 <+7>: push DWORD PTR [ecx-0x4]
0x08049195 <+10>: push ebp
0x08049196 <+11>: mov ebp,esp
0x08049198 <+13>: push ebx
0x08049199 <+14>: push ecx
0x0804919a <+15>: sub esp,0x10
0x0804919d <+18>: call 0x080490a0 <__x86.get_pc_thunk.bx>
0x080491a2 <+23>: add ebx,0x2e5e
0x080491a8 <+29>: mov DWORD PTR [ebp-0xc],0x5
0x080491af <+36>: mov DWORD PTR [ebp-0x10],0x6
0x080491b6 <+43>: push DWORD PTR [ebp-0x10]
0x080491b9 <+46>: push DWORD PTR [ebp-0xc]
0x080491bc <+49>: call 0x08049162 <add_plus1>
0x080491c1 <+54>: add esp,0x8
0x080491c4 <+57>: mov DWORD PTR [ebp-0x14],eax
0x080491c7 <+60>: sub esp,0x8
0x080491ca <+63>: push DWORD PTR [ebp-0x14]
0x080491cd <+66>: lea eax,[ebx-0x1ff8]
0x080491d3 <+72>: push eax
0x080491d4 <+73>: call 0x08049030 <printf@plt>
0x080491d9 <+78>: add esp,0x10
0x080491dc <+81>: mov eax,0x0
0x080491e1 <+86>: push [ebp-0x8]
0x080491e4 <+89>: pop ecx
0x080491e5 <+90>: pop ebx
0x080491e6 <+91>: pop ebp
0x080491e7 <+92>: lea esp,[ecx-0x4]
0x080491ea <+95>: ret
End of assembler dump.
```

Source Image 1

```
0xffffd6a8 +0x0000: 0x00000005 - $esp
0xffffd6ac +0x0004: 0x00000006
0xffffd6b0 +0x0008: 0x00000001
0xffffd6b4 +0x000c: 0xffffd784 - 0xffffd8a3 - "/workdir/lab1"
0xffffd6b8 +0x0010: 0x00000006
0xffffd6bc +0x0014: 0x00000005
0xffffd6c0 +0x0018: 0xffffd6e9 - 0x00000001
0xffffd6c4 +0x001c: 0x00000000

code:x86:32
0x080491af <main+36> mov DWORD PTR [ebp-0x10], 0x6
0x080491b6 <main+43> push DWORD PTR [ebp-0x10]
0x080491b9 <main+46> push DWORD PTR [ebp-0xc]
-> 0x080491bc <main+49> call 0x08049162 <add_plus1>
   0x08049162 <add_plus1+0> push ebp
   0x08049163 <add_plus1+1> mov ebp, esp
   0x08049165 <add_plus1+3> sub esp, 0x10
   0x08049168 <add_plus1+6> call 0x080491eb <__x86.get_pc_thunk.ax>
   0x0804916d <add_plus1+11> add eax, 0x2e93
   0x08049172 <add_plus1+16> mov eax, DWORD PTR [ebp+0x8]
                                arguments (guessed)

add_plus1 (
[sp + 0x0] = 0x00000005,
[sp + 0x4] = 0x00000006,
[sp + 0x8] = 0x00000001,
[sp + 0xc] = 0xffffd784 - 0xffffd8a3 - "/workdir/lab1",
[sp + 0x10] = 0x00000006
)

threads
[#0] Id 1, Name: "lab1", stopped 0x080491bc in main (), reason: SINGLE STEP
trace
[#0] 0x080491bc -> main()
```

Source Image 2

```
[ Legend: Modified register | Code | Heap | Stack | String ]

$eax : 0xc
$ebx : 0x0804c000 -> 0x0804bf14 -> 0x00000001
$ecx : 0xffffd6e0 -> 0x00000001
$edx : 0x5
$esp : 0xffffd690 -> 0xf7fb0000 -> 0x001e9d6c
$ebp : 0xffffd6a0 -> 0xffffd6c8 -> 0x00000000
$esi : 0xf7fb0000 -> 0x001e9d6c
$edi : 0xf7fb0000 -> 0x001e9d6c
$eip : 0x08049189 -> <add_plus1+39> leave
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0023 $ss: 0x002b $ds: 0x002b $es: 0x002b $fs: 0x0000 $gs: 0x0063

0xffffd690 +0x0000: 0xf7fb0000 -> 0x001e9d6c - $esp
0xffffd694 +0x0004: 0xf7fe2280 -> push ebp
0xffffd698 +0x0008: 0x00000006
0xffffd69c +0x000c: 0x00000005
0xffffd6a0 +0x0010: 0xffffd6c8 -> 0x00000000 - $ebp
0xffffd6a4 +0x0014: 0x080491c1 -> <main+54> add esp, 0x8
0xffffd6a8 +0x0018: 0x00000005
0xffffd6ac +0x001c: 0x00000006

0x08049181 <add_plus1+31> mov eax, DWORD PTR [ebp-0x8]
0x08049184 <add_plus1+34> add eax, edx
0x08049186 <add_plus1+36> add eax, 0x1
-> 0x08049189 <add_plus1+39> leave
0x0804918a <add_plus1+40> ret
0x0804918b <main+0> lea ecx, [esp+0x4]
0x0804918f <main+4> and esp, 0xffffffff
0x08049192 <main+7> push DWORD PTR [ecx-0x4]
0x08049195 <main+10> push ebp
```

Source Image 3

