

PSP0201

Week 5

Writeup

Group Name: Undecided

Members

ID	Name	Role
1211101390	Aslamia Najwa Binti Ahmad Khadri	Leader
1211100431	Mohammad Omar Torofder	Member
1211103388	Vishnu Karmegam	Member
1211103092	Farryn Aisha binti Muhd Firdaus	Member

[Day 17: Reverse Engineering – ReverseELFneering](#)

Tools used: Attackbox, Firefox, Virtualbox

Solution/walkthrough:

Question 1

View the pdf@main and print the code that is in the sim.main

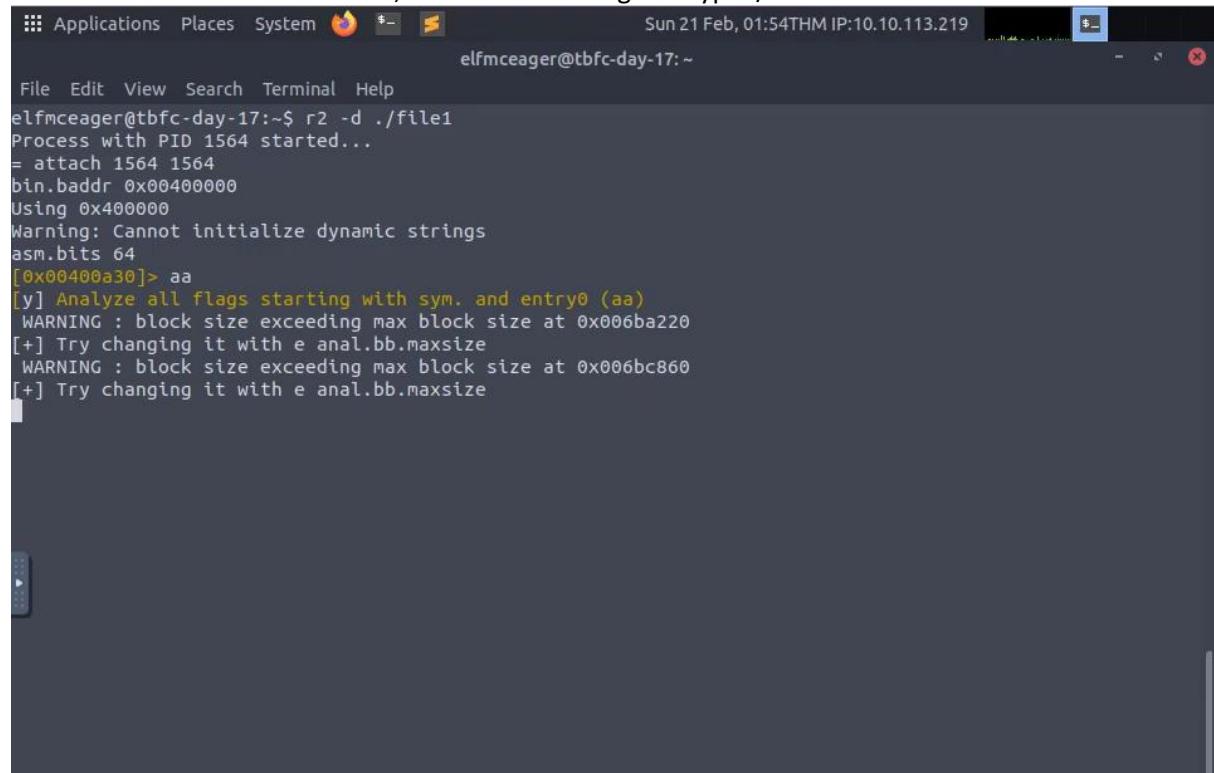
A register is a workbench where we can work on task one at a time where we can restore its value. We will have a register that will hold the value of local ch and another folder that hold the local 8h.

We have to perform mathematical functions on the two registers and it will overwrite the source register with the result.

Therefore, the data type that were matched with the size in bytes is byte = 1, Word = 2, Double Word = 4, Quad = 8, Single Precision = 4, Double Precision = 8

Question 2

We have to run file 1 and attach, r2 -d means debug and type ./file1.



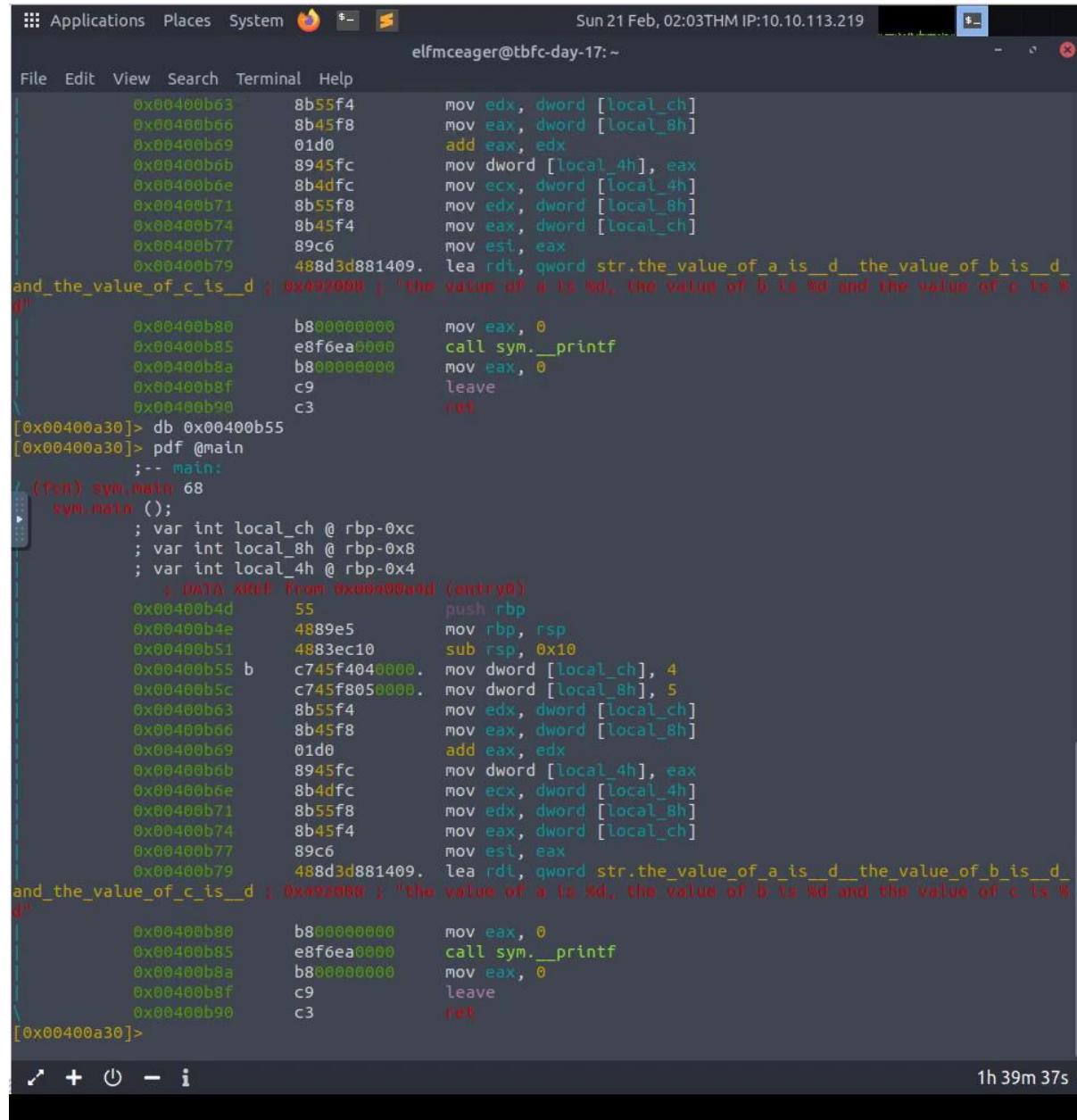
The screenshot shows a terminal window titled "elfmceager@tbfc-day-17:~". The window has a dark theme with light-colored text. At the top, there's a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". Below the menu, the command `r2 -d ./file1` is entered, followed by the output of the debugger. The output shows the debugger attaching to a process with PID 1564, setting the binary address to 0x00400000, and using 64-bit assembly. It then prompts for analysis with the command "[0x00400a30]> aa". The terminal also displays several warning messages about block sizes exceeding max block size at various addresses (0x006ba220, 0x006bc860) and suggests changing the analysis settings with "e anal.bb.maxsize".

```
elfmceager@tbfc-day-17:~$ r2 -d ./file1
Process with PID 1564 started...
= attach 1564 1564
bin.baddr 0x00400000
Using 0x400000
Warning: Cannot initialize dynamic strings
asm.bits 64
[0x00400a30]> aa
[y] Analyze all flags starting with sym. and entry0 (aa)
WARNING : block size exceeding max block size at 0x006ba220
[+] Try changing it with e anal.bb.maxsize
WARNING : block size exceeding max block size at 0x006bc860
[+] Try changing it with e anal.bb.maxsize
```

Thus, aa is the command that we entered to start analysing.

Question 3

We can set the breakpoint by using the db command. Type db0x00400b55 and pdf@main.



The screenshot shows a GDB session in a terminal window. The assembly code for the main function is displayed, showing variable declarations and their addresses. A breakpoint is set at address 0x00400b55, indicated by a lowercase 'b' prefix in the assembly listing. The command 'pdf @main' is used to print the registers and memory dump at the start of the main function. The assembly code includes instructions for initializing local variables (local_ch, local_8h, local_4h) and printing them using the printf function.

```
[0x00400a30]> db 0x00400b55
[0x00400a30]> pdf @main
;; main:
> (tcb) sym.main 68
sym.main ()@main:
    ; var int local_ch @ rbp-0xc
    ; var int local_8h @ rbp-0x8
    ; var int local_4h @ rbp-0x4
        ; DATA AT&T From 0x00400a4d (entry4)
0x00400b4d      55          push rbp
0x00400b4e      4889e5      mov rbp, rsp
0x00400b51      4883ec10    sub rsp, 0x10
0x00400b55 b   c745f4040000. mov dword [local_ch], 4
0x00400b5c      c745f8050000. mov dword [local_8h], 5
0x00400b63      8b55f4      mov edx, dword [local_ch]
0x00400b66      8b45f8      mov eax, dword [local_8h]
0x00400b69      01d0        add eax, edx
0x00400b6b      8945fc      mov dword [local_4h], eax
0x00400b6e      8b4dfc      mov ecx, dword [local_4h]
0x00400b71      8b55f8      mov edx, dword [local_8h]
0x00400b74      8b45f4      mov eax, dword [local_ch]
0x00400b77      89c6        mov esi, eax
0x00400b79      488d3d881409. lea rdi, qword str.the_value_of_a_is_d_the_value_of_b_is_d_
and_the_value_of_c_is_d ; 0x402000 ; "the value of a is %d, the value of b is %d and the value of c is %d"
0x00400b80      b800000000  mov eax, 0
0x00400b85      e8f6ea0000  call sym.__printf
0x00400b8a      b800000000  mov eax, 0
0x00400b8f      c9          leave
0x00400b90      c3          ret
[0x00400a30]>
```

We will see a lowercase b next to it which is telling us that we have a breakpoint.

Question 4

We can type in dc which starts sunning the code and it will run up the first breakpoint.

The screenshot shows a terminal window with the following content:

```
Sun 21 Feb, 02:04THM IP:10.10.113.219
elfmceager@tbfc-day-17: ~

File Edit View Search Terminal Help
0x00400b69    01d0      add eax, edx
0x00400b6b    8945fc    mov dword [local_4h], eax
0x00400b6e    8b4dfc    mov ecx, dword [local_4h]
0x00400b71    8b55f8    mov edx, dword [local_8h]
0x00400b74    8b45f4    mov eax, dword [local_ch]
0x00400b77    89c6      mov est, eax
0x00400b79    488d3d881409. lea rdi, qword str.the_value_of_a_is_d_the_value_of_b_is_d_
and_the_value_of_c_is_d ; 0x402008 ; "the value of a is %d, the value of b is %d and the value of c is %d
0x00400b80    b800000000  mov eax, 0
0x00400b85    e8f0ea0000  call sym._printf
0x00400b8a    b800000000  mov eax, 0
0x00400b8f    c9        leave
0x00400b90    c3        ret

[0x00400a30]> db 0x00400b55
[0x00400a30]> pdf @main
-- main:
/(Fon) sym.main 68
sym.main ();
; var int local_ch @ rbp-0xc
; var int local_8h @ rbp-0x8
; var int local_4h @ rbp-0x4
    j DATA XREF from 0x00000000 (entryp)
0x00400b4d    55        push rbp
0x00400b4e    4889e5    mov rbp, rsp
0x00400b51    4883ec10  sub rsp, 0x10
0x00400b55 b   c745f4040000. mov dword [local_ch], 4
0x00400b5c    c745f8050000. mov dword [local_8h], 5
0x00400b63    8b55f4    mov edx, dword [local_ch]
0x00400b66    8b45f8    mov eax, dword [local_8h]
0x00400b69    01d0      add eax, edx
0x00400b6b    8945fc    mov dword [local_4h], eax
0x00400b6e    8b4dfc    mov ecx, dword [local_4h]
0x00400b71    8b55f8    mov edx, dword [local_8h]
0x00400b74    8b45f4    mov eax, dword [local_ch]
0x00400b77    89c6      mov est, eax
0x00400b79    488d3d881409. lea rdi, qword str.the_value_of_a_is_d_the_value_of_b_is_d_
and_the_value_of_c_is_d ; 0x402008 ; "the value of a is %d, the value of b is %d and the value of c is %d
0x00400b80    b800000000  mov eax, 0
0x00400b85    e8f0ea0000  call sym._printf
0x00400b8a    b800000000  mov eax, 0
0x00400b8f    c9        leave
0x00400b90    c3        ret

[@0x00400a30]> dc
hit breakpoint at: 400b55
[0x00400b55]>
```

The terminal window has a title bar "elfmceager@tbfc-day-17: ~" and a status bar "1h 38m 45s". The bottom of the window shows a toolbar with icons for file operations.

We have set a memory location 400b55 and that is where it stopped. When we do a pdf@main again, we will see it is highlighted and there is a rip comment where we are currently sitting.

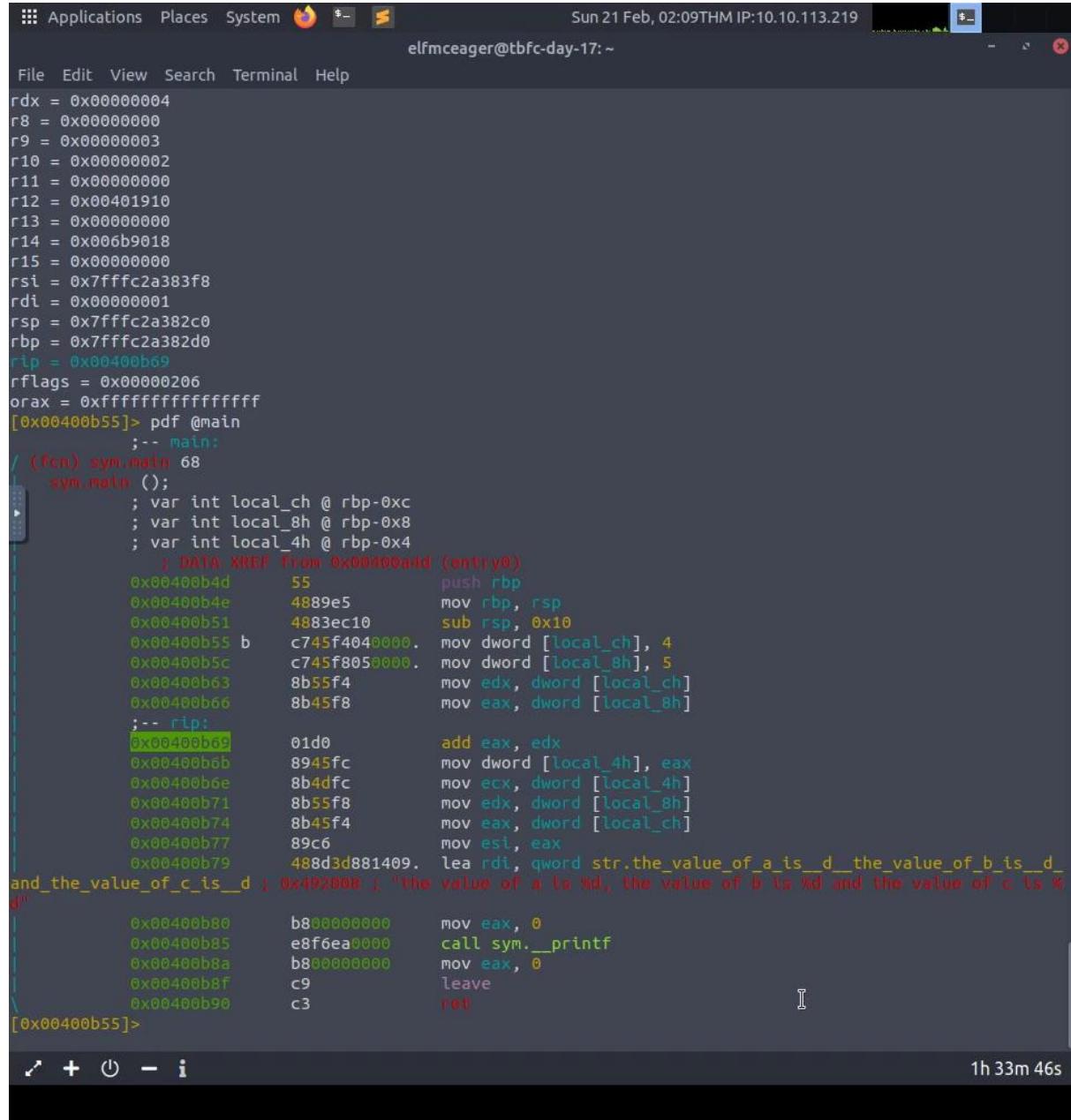
The screenshot shows the Immunity Debugger interface. The assembly pane displays the following code:

```
File Edit View Terminal Help
elfmceager@tbfc-day-17:~ Sun 21 Feb, 02:04THM IP:10.10.113.219
0x00400b6b    8945fc      mov dword [local_4h], eax
0x00400b6e    8b4dfc      mov ecx, dword [local_4h]
0x00400b71    8b55f8      mov edx, dword [local_8h]
0x00400b74    8b45f4      mov eax, dword [local_ch]
0x00400b77    89c6        mov esi, eax
0x00400b79    488d3d881409. lea rdi, qword str.the_value_of_a_is_d_the_value_of_b_is_d_
and_the_value_of_c_is_d ; 0x492908 ; "the value of a is %d, the value of b is %d and the value of c is %d"
0x00400b80    b800000000  mov eax, 0
0x00400b85    e8f6ea0000  call sym.__printf
0x00400b8a    b800000000  mov eax, 0
0x00400b8f    c9          leave
0x00400b90    c3          ret
[0x00400a30]> dc
hit breakpoint at: 400b55
[0x00400b55]> pdf @main
;-- main:
;-- fax:
/ (Func) sym.main 68
sym.main ();
    ; var int local_ch @ rbp-0xc
    ; var int local_8h @ rbp-0x8
    ; var int local_4h @ rbp-0x4
    ; DATA XREF From 0x00400a4d (entry0)
0x00400b4d    55          push rbp
0x00400b4e    4889e5      mov rbp, rsp
0x00400b51    4883ec10    sub rsp, 0x10
;-- rip:
0x00400b55 b   c745f4040000. mov dword [local_ch], 4
0x00400b5c    c745f8050000. mov dword [local_8h], 5
0x00400b63    8b55f4      mov edx, dword [local_ch]
0x00400b66    8b45f8      mov eax, dword [local_8h]
0x00400b69    01d0        add eax, edx
0x00400b6b    8945fc      mov dword [local_4h], eax
0x00400b6e    8b4dfc      mov ecx, dword [local_4h]
0x00400b71    8b55f8      mov edx, dword [local_8h]
0x00400b74    8b45f4      mov eax, dword [local_ch]
0x00400b77    89c6        mov esi, eax
0x00400b79    488d3d881409. lea rdi, qword str.the_value_of_a_is_d_the_value_of_b_is_d_
and_the_value_of_c_is_d ; 0x492908 ; "the value of a is %d, the value of b is %d and the value of c is %d"
0x00400b80    b800000000  mov eax, 0
0x00400b85    e8f6ea0000  call sym.__printf
0x00400b8a    b800000000  mov eax, 0
0x00400b8f    c9          leave
0x00400b90    c3          ret
[0x00400b55]>
```

The assembly code is identical to the previous screenshot, but the instruction at address 0x00400b55 is highlighted in green, indicating it is the current instruction being executed. The rip comment at the bottom of the assembly pane also highlights this instruction.

Question 5,6 & 7

When we look at the code again, we will see that we are right at the line where we do the addition of eax + edx



The screenshot shows the GDB debugger interface on a Linux system. The title bar indicates the date and IP address: Sun 21 Feb, 02:09THM IP:10.10.113.219. The command line shows the current address is 0x00400b55, pointing to the start of the main function. The assembly code for main starts with a prologue (push rbp, mov rbp, rsp), initializes local variables (local_ch, local_8h, local_4h), and then enters a loop. Inside the loop, it adds the values of rdx and rax (add eax, edx) and stores the result in eax. The code then prints the values of eax, b, and c using printf. The assembly code is color-coded by the debugger to highlight different instructions and registers.

```
rdx = 0x00000004
r8 = 0x00000000
r9 = 0x00000003
r10 = 0x00000002
r11 = 0x00000000
r12 = 0x00401910
r13 = 0x00000000
r14 = 0x006b9018
r15 = 0x00000000
rst = 0x7fffcc2a383f8
rdi = 0x00000001
rsp = 0x7fffcc2a382c0
rbp = 0x7fffcc2a382d0
rip = 0x00400b59
rflags = 0x000000206
orax = 0xfffffffffffffff
[0x00400b55]> pdf @main
;-- main:
/ (fcn) sym.main 68
sym.main ()@main
    ; var int local_ch @ rbp-0xc
    ; var int local_8h @ rbp-0x8
    ; var int local_4h @ rbp-0x4
    ; DATA XREF From 0x00100add (entry/y0)
0x00400b4d      55          push rbp
0x00400b4e      4889e5      mov rbp, rsp
0x00400b51      4883ec10   sub rsp, 0x10
0x00400b55 b    c745f4040000. mov dword [local_ch], 4
0x00400b5c      c745f8050000. mov dword [local_8h], 5
0x00400b63      8b45f4      mov edx, dword [local_ch]
0x00400b66      8b45f8      mov eax, dword [local_8h]
;-- rip:
0x00400b69      01d0        add eax, edx
0x00400b6b      8945fc      mov dword [local_4h], eax
0x00400b6e      8b4dfc      mov ecx, dword [local_4h]
0x00400b71      8b55f8      mov edx, dword [local_8h]
0x00400b74      8b45f4      mov eax, dword [local_ch]
0x00400b77      89c6        mov esi, eax
0x00400b79      488d3d881409. lea rdi, qword str.the_value_of_a_is_d_the_value_of_b_is_d_and_the_value_of_c_is_d ; 0x402008 ; 'the value of a is %d, the value of b is %d and the value of c is %d'
and_the_value_of_c_is_d ; 0x402008 ; 'the value of a is %d, the value of b is %d and the value of c is %d'
0x00400b80      b800000000  mov eax, 0
0x00400b85      e8f6ea0000  call sym.__printf
0x00400b8a      b800000000  mov eax, 0
0x00400b8f      c9          leave
0x00400b90      c3          ret
[0x00400b55]>
```

We have to look at the registers, $rax = 5$, $rdx = 4$, when we step, we will perform the addition and one of our registers will be modified.

The screenshot shows a terminal window with the following content:

```

[0x004000b55]> pdf @main
;-- main:
/ (Fcn) sym.main 68
sym.main () {
    ; var int local_ch @ rbp-0xc
    ; var int local_8h @ rbp-0x8
    ; var int local_4h @ rbp-0x4
    ; DATA XREF From 0x0040004d (entry0)
    0x004000b4d      55          push rbp
    0x004000b4e      4899e5     mov rbp, rsp
    0x004000b51      4883ec10  sub rsp, 0x10
    0x004000b55 b    c745f4040000. mov dword [local_ch], 4
    0x004000b5c      c745f8050000. mov dword [local_8h], 5
    0x004000b63      8b55f4     mov edx, dword [local_ch]
    0x004000b66      8b45f8     mov eax, dword [local_8h]
    ;-- rip:
    0x004000b69      01d0       add eax, edx
    0x004000b6b      8945fc     mov dword [local_4h], eax
    0x004000b6e      8b4dfc     mov ecx, dword [local_4h]
    0x004000b71      8b55f8     mov edx, dword [local_8h]
    0x004000b74      8b45f4     mov eax, dword [local_ch]
    0x004000b77      89c6       mov est, eax
    0x004000b79      488d3d881409. lea rdi, qword str.the_value_of_a_is_d_the_value_of_b_is_d_
and_the_value_of_c_is_d ; 0x492008 ; "the value of a is %d, the value of b is %d and the value of c is %d"
    0x004000b80      b800000000  mov eax, 0
    0x004000b85      e8f6ea0000  call sym._printf
    0x004000b8a      b800000000  mov eax, 0
    0x004000b8f      c9         leave
    0x004000b90      c3         ret
[0x004000b55]> dr
rax = 0x00000005
rbx = 0x00400400
rcx = 0x0044ba90
rdx = 0x00000004
r8 = 0x00000000
r9 = 0x00000003
r10 = 0x00000002
r11 = 0x00000000
r12 = 0x000401910
r13 = 0x00000000
r14 = 0x006b9018
r15 = 0x00000000
rsi = 0x7fff2a383f8
rdt = 0x00000001
rsp = 0x7fff2a382c0
rbp = 0x7fff2a382d0
rbp = 0x004000b69

```

Registers (dr output):

- rax = 0x00000005
- rbx = 0x00400400
- rcx = 0x0044ba90
- rdx = 0x00000004
- r8 = 0x00000000
- r9 = 0x00000003
- r10 = 0x00000002
- r11 = 0x00000000
- r12 = 0x000401910
- r13 = 0x00000000
- r14 = 0x006b9018
- r15 = 0x00000000
- rsi = 0x7fff2a383f8
- rdt = 0x00000001
- rsp = 0x7fff2a382c0
- rbp = 0x7fff2a382d0
- rbp = 0x004000b69

Bottom status bar: 1h 33m 12s

Thus, 1 is the value of local_ch when its corresponding movl instruction is called, 6 is the value of eax when the imull instruction is called, and 6 is the value of local_4h before eax is set to 0.

Thought Process/Methodology:

Print the code found in sim.main and view the pdf@main. A register serves as a work surface where we can do tasks one at a time in order to recover their value. The local ch value will be kept in a register, while the local 8h value will be kept in another register. The two registers must be subjected to mathematical operations, and the results will be written over the source register. As a result, byte = 1, word = 2, double word = 4, quad = 8, single precision = 4, double precision = 8, and the data type were matched with the size in bytes. To execute file 1 and attach, type ./file1. Type r2 -d to debug. Thus, we entered the command aa to begin our analysis. Using the db command, we can set the breakpoint. Enter pdf@main and db0x00400b55. The fact that there is a lowercase b next to it indicates that we have reached a breakpoint. We may start the code by typing dc, and it will run up

the first breakpoint. It halted at a memory location that we placed at 400b55. It will be underlined and have a rip comment where we are sitting if we run a pdf@main once more. Rereading the code will reveal that we are currently on the line where the addition of eax and edx is performed. When we step, we will conduct the addition, and one of our registers will be changed. The registers are: rax = 5, rdx = 4, etc. Thus, 1 represents the value of local ch when the movl instruction corresponding to it is called, 6 represents the value of eax when the imull instruction is executed, and 6 represents the value of local 4h prior to eax being set to 0.