PSP0201 Week 5 Writeup

Group Name: ikun no 1

Members

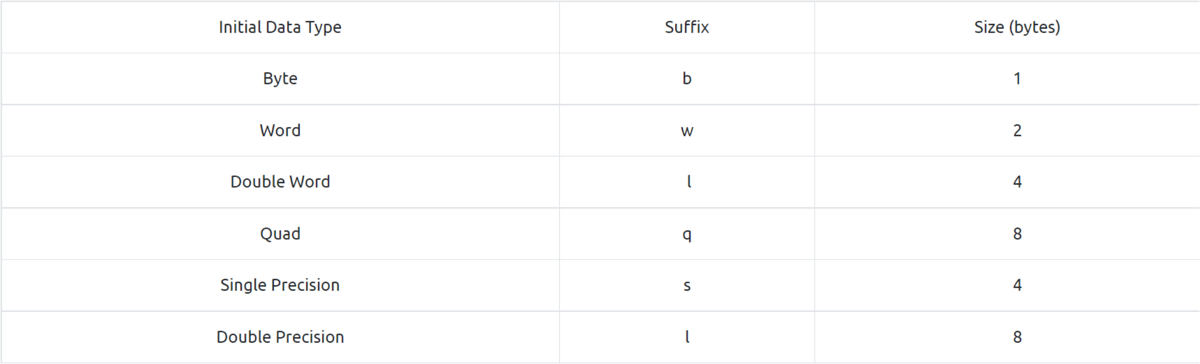
ID	Name	Role
1211102058	Chu Liang Chern	Leader
1211101401	Chong Jii Hong	Member
1211103206	Ng Kai Keat	Member
1211103095	Siddiq Ferhad Bin Khairil Anual	Member

Day 17 - [Reverse Engineering] ReverseELFneering

Tool used: kali Linux, Firefox

Solution/Walkthrough:

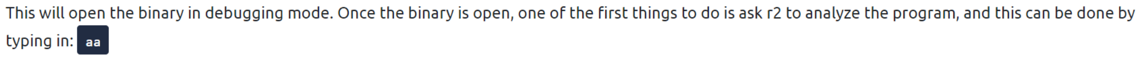
Q1.



Initial Data Type	Suffix	Size (bytes)
Byte	b	1
Word	w	2
Double Word	l	4
Quad	q	8
Single Precision	s	4
Double Precision	l	8

The answer can be found and taken from THM.

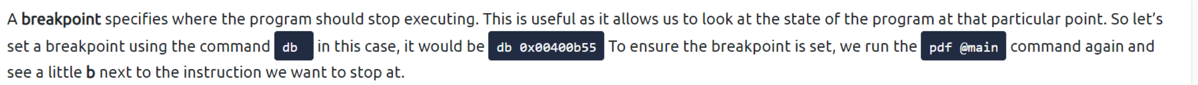
Q2.



This will open the binary in debugging mode. Once the binary is open, one of the first things to do is ask r2 to analyze the program, and this can be done by typing in: `aa`

The command aa can be used to analyse the programs in radare2.

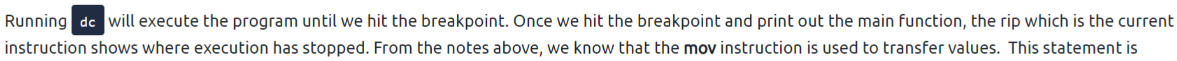
Q3.



A **breakpoint** specifies where the program should stop executing. This is useful as it allows us to look at the state of the program at that particular point. So let's set a breakpoint using the command `db` in this case, it would be `db 0x00400b55` To ensure the breakpoint is set, we run the `pdf @main` command again and see a little **b** next to the instruction we want to stop at.

By using the db command, we can set a breakpoint in radare2..

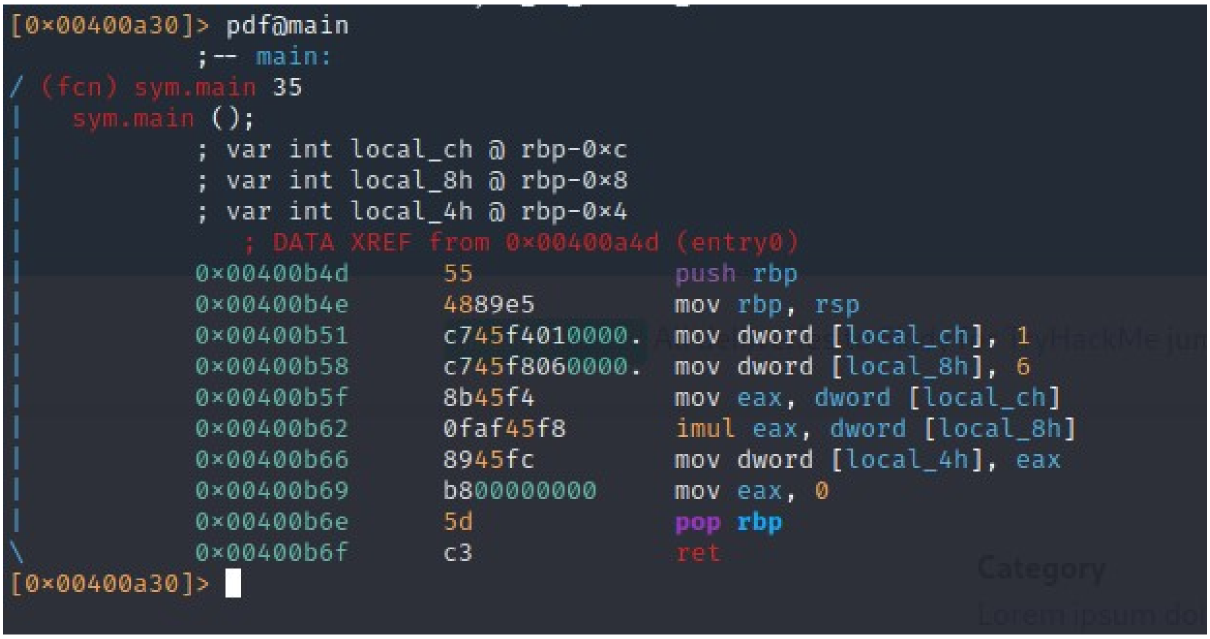
Q4.



Running `dc` will execute the program until we hit the breakpoint. Once we hit the breakpoint and print out the main function, the rip which is the current instruction shows where execution has stopped. From the notes above, we know that the **mov** instruction is used to transfer values. This statement is

As stated in THM, dc can be used to execute the program until we hit the breakpoint.

Q5,Q6,Q7.



```
[0x00400a30]> pdf@main
            ;-- main:
/ (fcn) sym.main 35
|   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      c745f4010000.  mov dword [local_ch], 1
|           0x00400b58      c745f8060000.  mov dword [local_8h], 6
|           0x00400b5f      8b45f4         mov eax, dword [local_ch]
|           0x00400b62      0faf45f8       imul eax, dword [local_8h]
|           0x00400b66      8945fc         mov dword [local_4h], eax
|           0x00400b69      b800000000     mov eax, 0
|           0x00400b6e      5d             pop rbp
\           0x00400b6f      c3             ret
[0x00400a30]> 
```

Q5. mov=move. Therefore, local _ch is 1.

Q6. imul= multiplications. local_ch is 1 and being moved to eax; eax =1.1 multiple by 6=6.

Q7. eax is 6. When eax is moved to local_4h, it became 6 too.