

| | | |
|--|------------|-------------------|
| FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY UNIVERSITI TEKNIKAL MALAYSIA MELAKA | | |
| MALWARE ANALYSIS & DIGITAL INVESTIGATION | | |
| BITS 3453 | SEMESTER I | SESSION 2022/2023 |



ASSIGNMENT 1

COURSE: BITZ

LECTURER: TS. DR. MOHD ZAKI BIN MAS'UD

GROUP MEMBERS:

| NAME | MATRIC NO. |
|---------------------------------|------------|
| NG WAN THONG | B032010104 |
| VINERD A/L MARIADASS | B032010206 |
| ZULIANA BINTI ZAKRI | B032010283 |
| MEIZA CERMELLA BINTI ABDUL AZIZ | B032010450 |

SECTION A: STATIC ANALYSIS

Tool used: Ghidra

- 1) Once Ghidra is installed, launch it using command **"./ghidraRun"** as shown in **Figure 1**

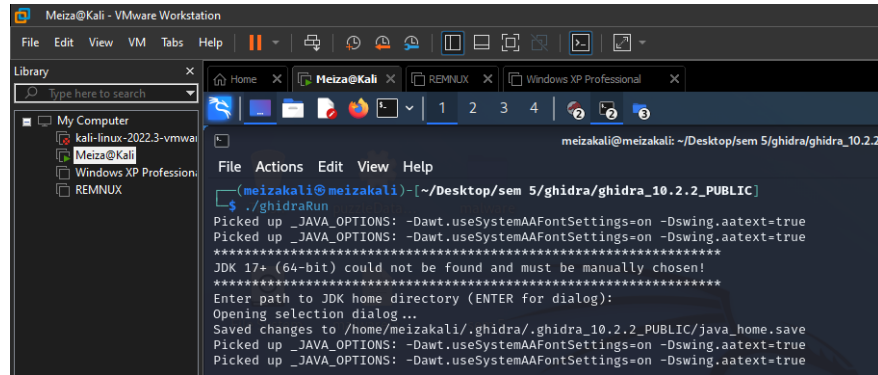


Figure 1 shows the command to launch the Ghidra

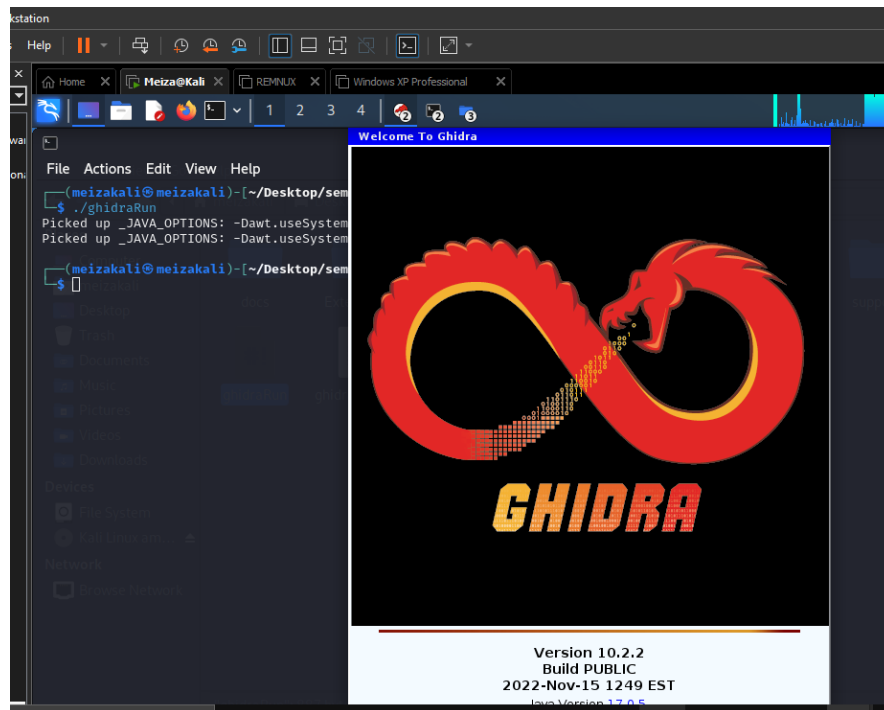


Figure 2 shows the startup page when launching the Ghidra

2) Click File, New File, and Select non-share project as shown in **Figure 3** and **4**.

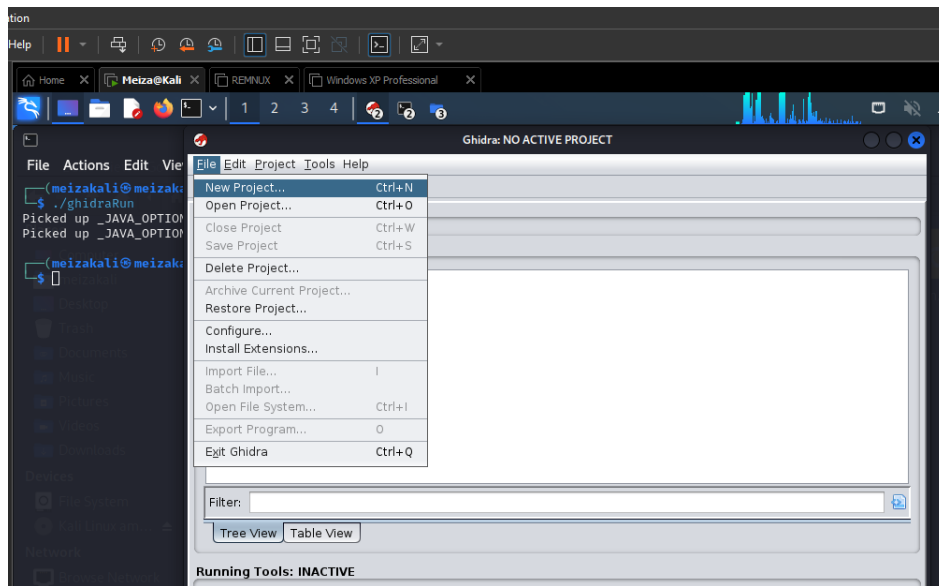


Figure 3

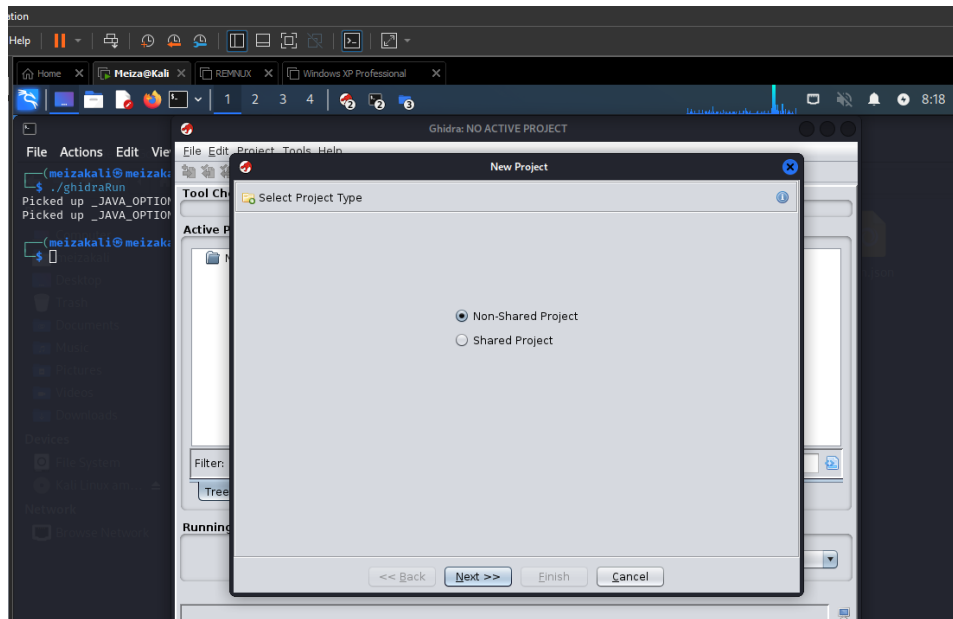


Figure 4

3) Input the Project Directory and Project Name

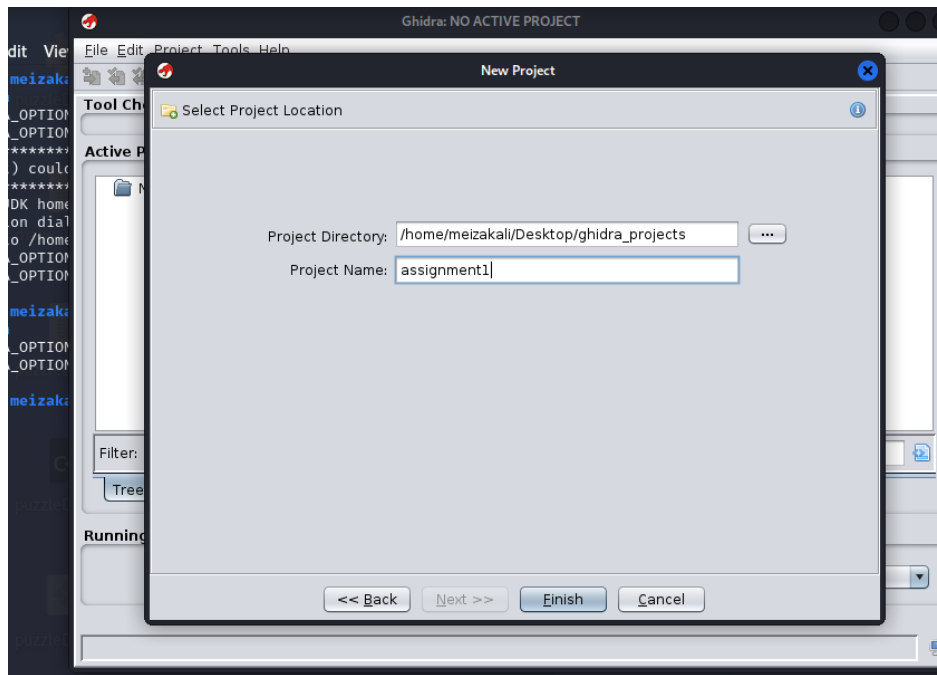


Figure 5

4) Import the binary file “puzzleFASA” which is the other group puzzle in the format of ELF

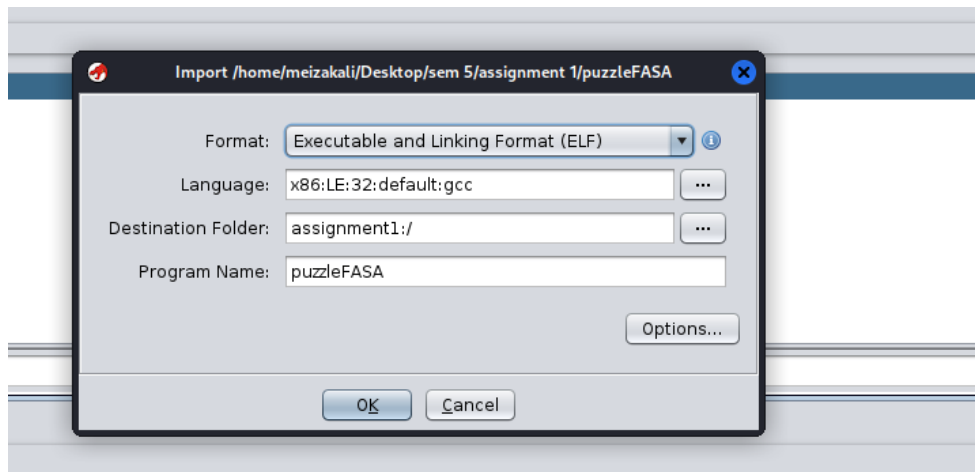


Figure 6

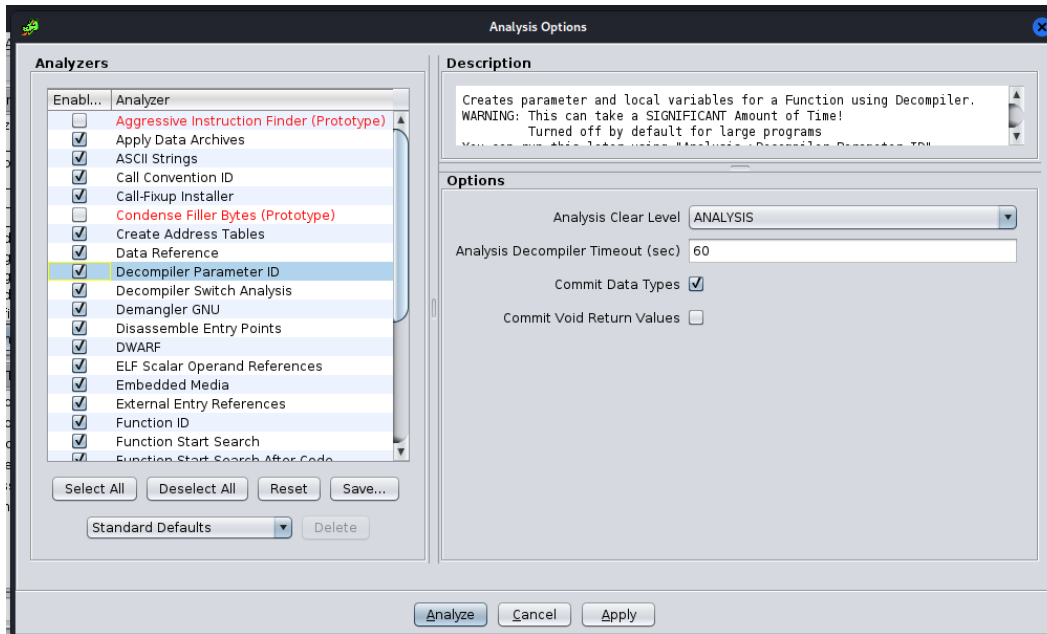


Figure 7 shows the Analysis options

- 5) The result from the analysis will show the **Decompile Window** that displays the decompilation result for the specific function selected in the listing window. Other than that, it also displays the **Listing Window** that shows the details of the memory address, opcode, and assembly code instruction from the analysis. **Program Tree Window** shows the segment of the ELF file, **Symbol Tree Window** lists and displays all currently defined symbols, **Data Type Manager Window** shows the data types inferred during the auto-analysis, and **Console Window** for the tool output.

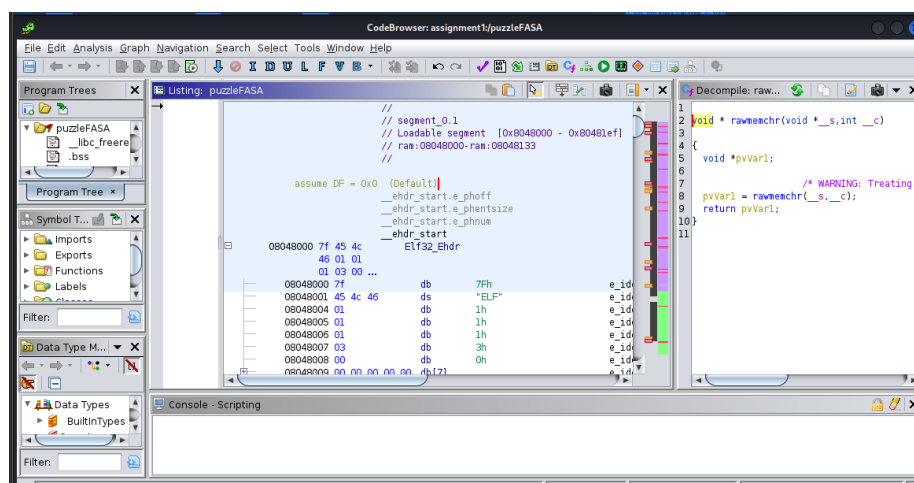


Figure 8

- 6) Look for the main function by typing “**main**” in the search bar under the section Symbol Tree window and it’ll display the details as shown in **Figure 9** below. The XREF is generated when Ghidra detects locations or instructions that points to the address shown below. When the function is not included in the **Formal Function Body**, the decompiler will invent it as **undefined x** where x can be 1,2,4, etc as the calling convention. In this case, the **undefine1** is declared as the unknown data-type of the **param_1**, while the **undefine4** is declare as the unknown data type of the main function.

Figure 9

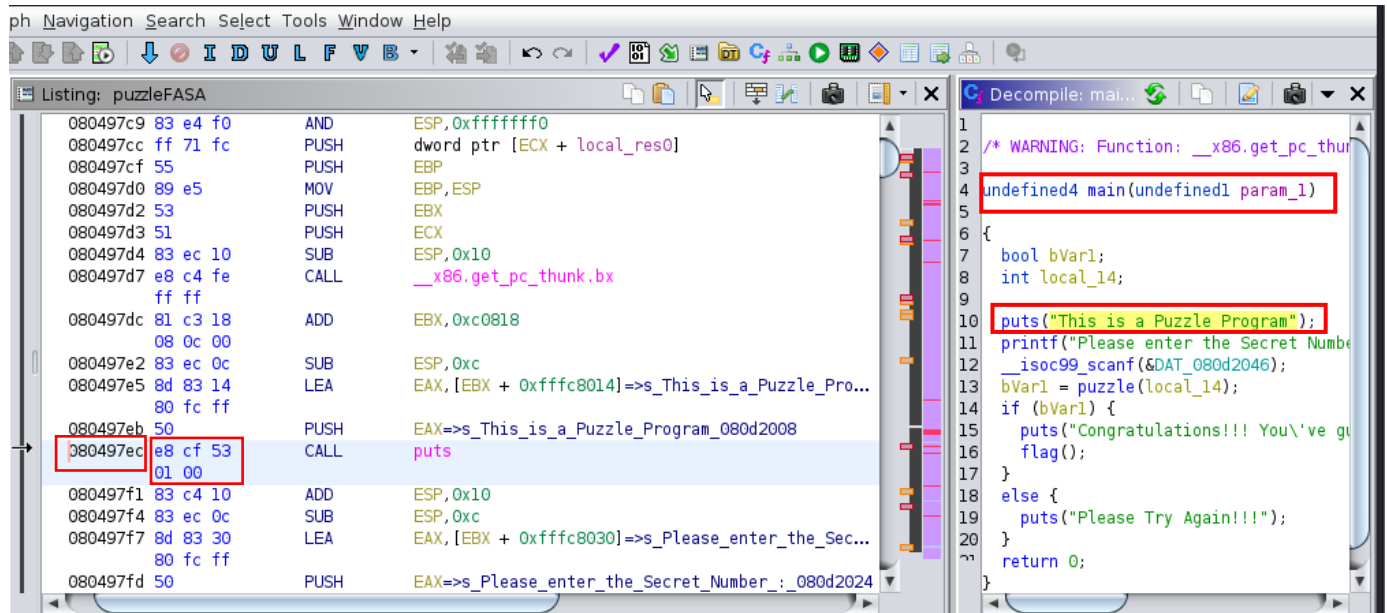


Figure 10

Based on **Figure 10**, in the decompilation window it displays the function at the current address **080497ec** as shown in the listing window. **080497ec** is the address field which represents the memory address where the data is located. The second column which is **e8 cf 53 01 00** represents the raw bytes which are the opcode of the instruction that's executed by the CPU. The third column shows the instruction, in this case is **CALL**. To interpret the code, it will call the predefined function **puts** to display the message "This is a Puzzle Program".

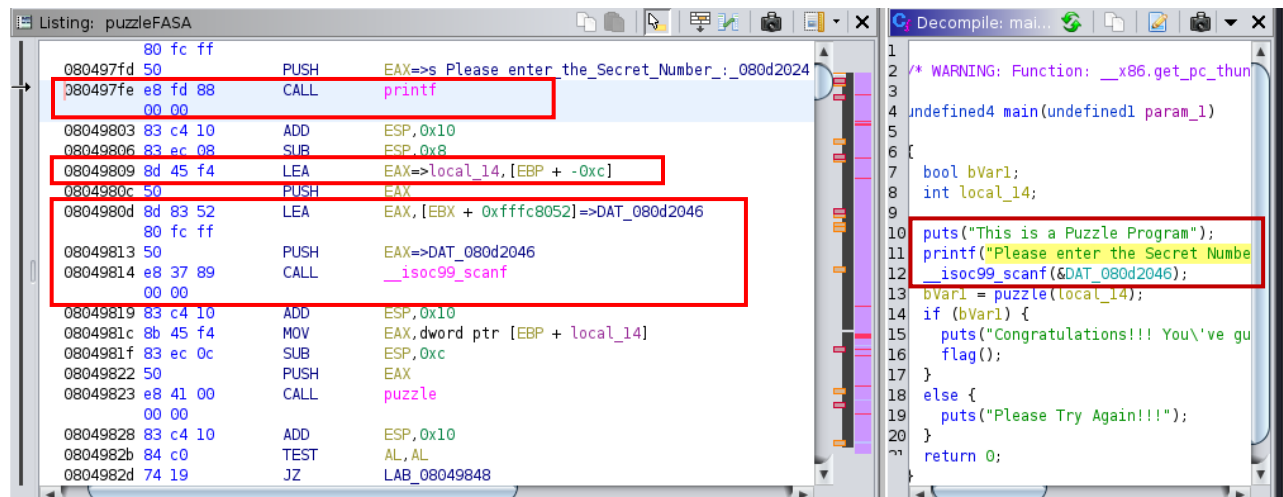


Figure 11

Based on **Figure 11**, it will call the function **printf** to display the message “**Please enter the Secret Number**”. The instruction **LEA** will store the second operand **EBX** referring **DAT_080d2046** into the first operand **EAX**. It will then **PUSH** the data of register **EAX** into the stack. Other than that, the instruction **CALL _isoc99_scanf** will return the user input value.

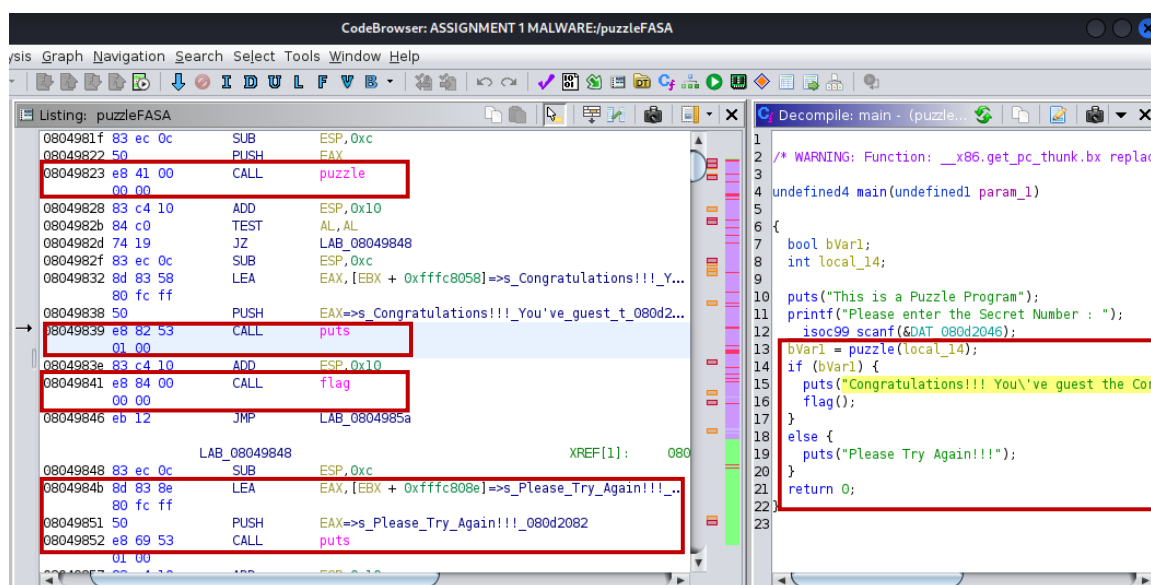


Figure 12

Based on **Figure 12**, the decompiler window shows the user input value will be tested inside the puzzle function when it passes the variable define as **local_14** with data type **int** as the parameter to the puzzle function. From the listing window, it will call the puzzle function at the memory address of **080498323**. If it's **true**, in this case define as **bVar1** then it will call the predefined function **puts** and display the message “**Congratulations!! You’ve guess the Correct Number!!!**” and it will call the **flag** at memory address **08049841**. Whereas if the condition is **not true**, it will call the function **puts** to display the message “**Please Try Again!!!**”.

- 7) On the Symbol Tree window, choose and click the menu **Functions > p > puzzle** to get the functions details as shown in **Figure 13**.

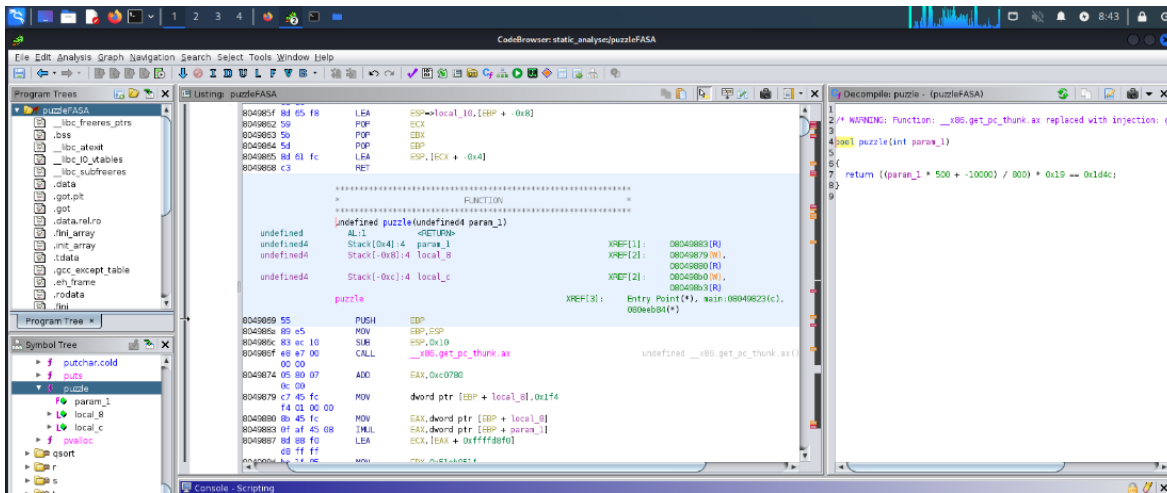


Figure 13

- 8) Calculate the secret number of **param_1** and get the answer as shown in **Figure 14**.

$$\text{Equation: } ((\text{param}_1 * 500 + -10000) / 800) * 0x19 == 0x1d4c$$

hexadecimal

Let:

$$\text{param}_1 = x$$

$$0x19 = 25$$

$$0x1d4c = 7500$$

Rewrite the equation:

$$((x * 500 + (-10000)) / 800) * 25 = 7500$$

$$(500x - 10000) / 800 = 300$$

$$500x - 10000 = 240000$$

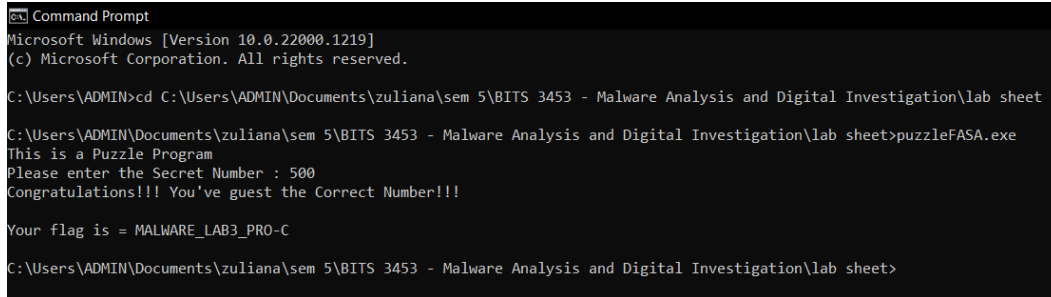
$$500x = 250000$$

$$x = 500$$

(secret number)

Figure 14

- 9) To run the puzzleFASA.exe file, open the command prompt on the host machine and change the directory to the directory that contains the puzzleFASA.exe file. Run the exe file by using the **puzzleFASA.exe** command. Put the answer that we get in **Figure 14**. All the steps are shown in **Figure 15** below.



```
Command Prompt
Microsoft Windows [Version 10.0.22000.1219]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>cd C:\Users\ADMIN\Documents\zuliana\sem 5\BITS 3453 - Malware Analysis and Digital Investigation\lab sheet

C:\Users\ADMIN\Documents\zuliana\sem 5\BITS 3453 - Malware Analysis and Digital Investigation\lab sheet>puzzleFASA.exe
This is a Puzzle Program
Please enter the Secret Number : 500
Congratulations!!! You've guest the Correct Number!!!

Your flag is = MALWARE_LAB3_PRO-C

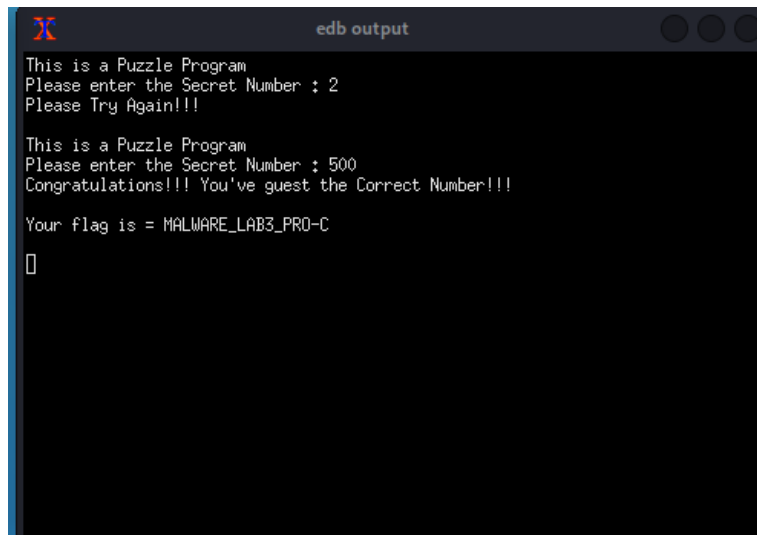
C:\Users\ADMIN\Documents\zuliana\sem 5\BITS 3453 - Malware Analysis and Digital Investigation\lab sheet>
```

Figure 15

SECTION B: DYNAMIC ANALYSIS

Tools used: EDB Debugger

1. Open edb-debugger.
2. Open file puzzleFASA.
3. Run puzzleFASA.
4. Fill in secret number and hit enter.
5. Try a wrong secret number and correct secret number obtained from static analysis.
6. Output shows 2 is not the correct answer while 500 is the correct answer for puzzle to be solved. (**Figure 16**)



```
edb output
This is a Puzzle Program
Please enter the Secret Number : 2
Please Try Again!!!

This is a Puzzle Program
Please enter the Secret Number : 500
Congratulations!!! You've guest the Correct Number!!!

Your flag is = MALWARE_LAB3_PRO-C
[]
```

Figure 16

7. Restart the puzzle program.
8. Add toggle breakpoint. (**Figure 17**)

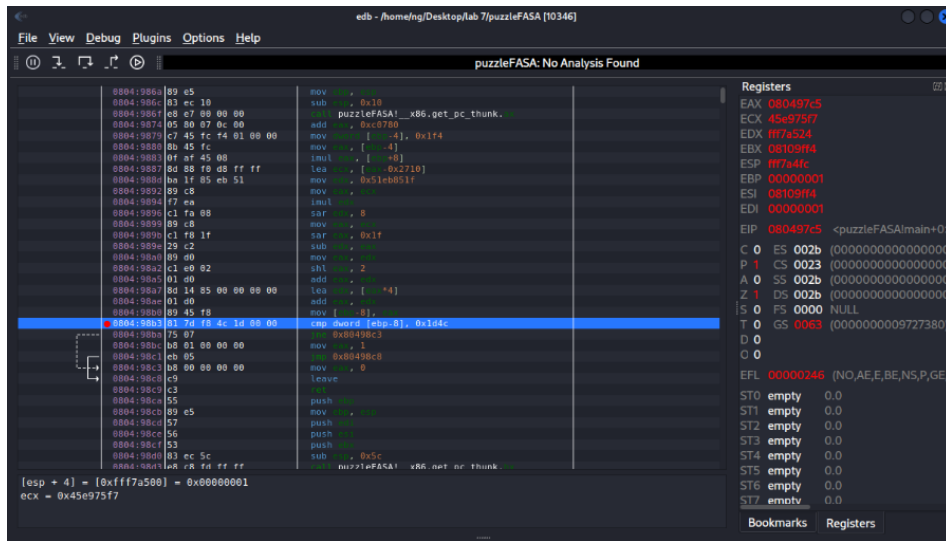


Figure 17

9. Run the puzzle.
10. Enter number 2 as the secret number again and hit enter.
11. Program paused at where the toggle breakpoint added. (**Figure 18**)

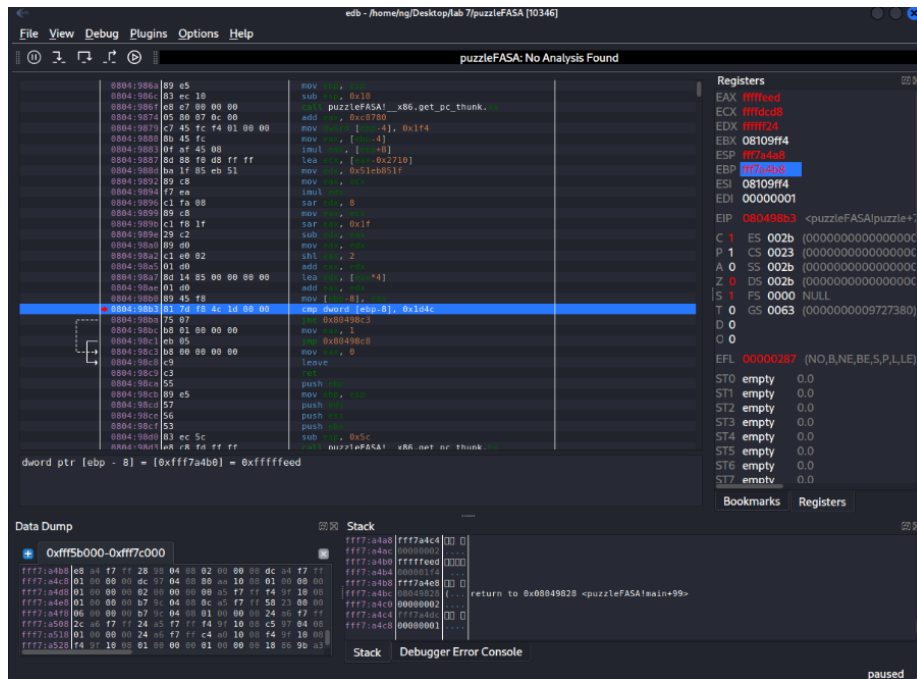


Figure 18

12. Right click on EBP on Registers panel and choose option “Follow in Dump”.
13. Under Data Dump panel, it shows that binary file value of [ebp-8] is 0xfeed. (**Figure 19**)

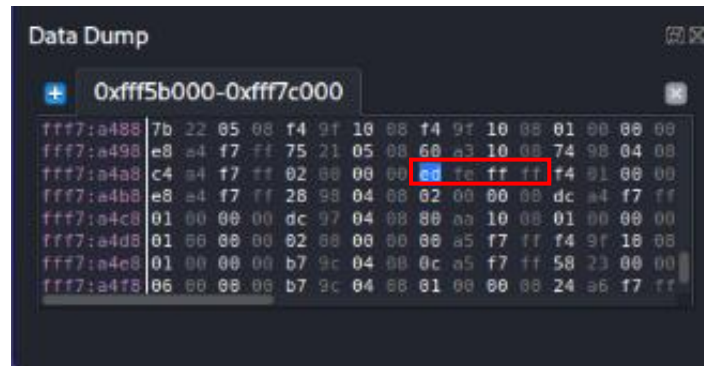


Figure 19

14. Right click on the binary string and choose “edit binary string”. (**Figure 20**)

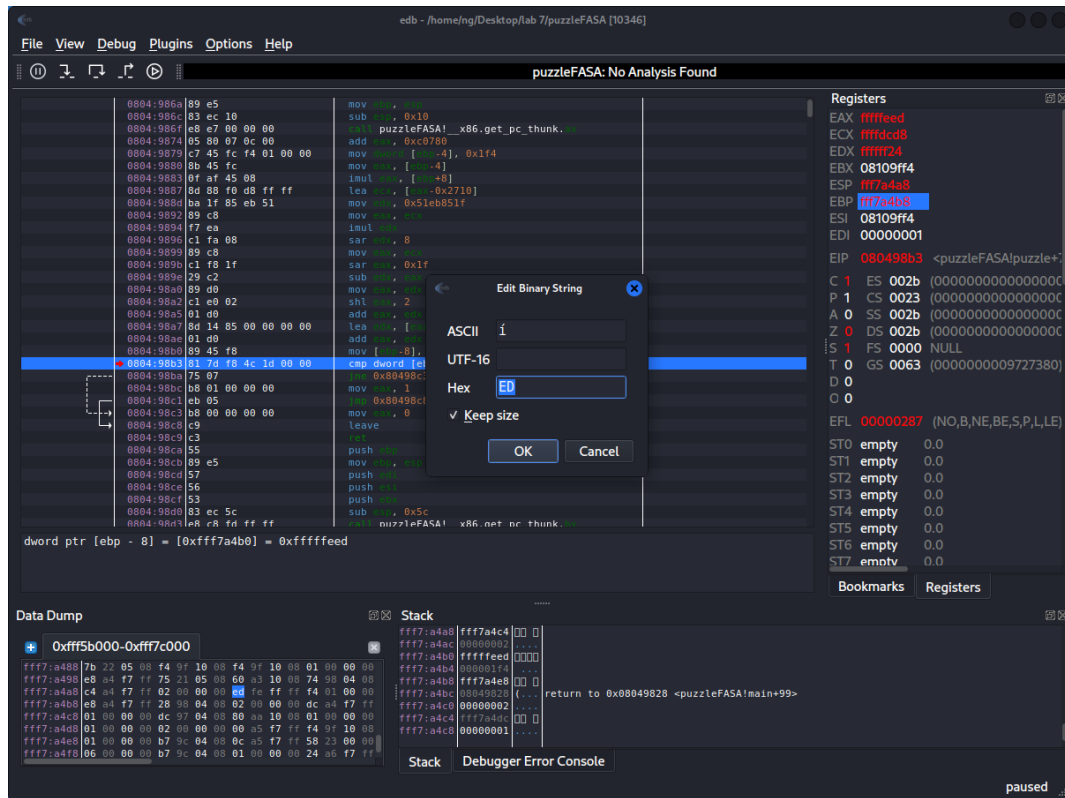


Figure 20

15. Change the value of binary string to 0x1d4c since the binary string of [ebp-8] will later be compared to 0x1d4c. (Figure 21)

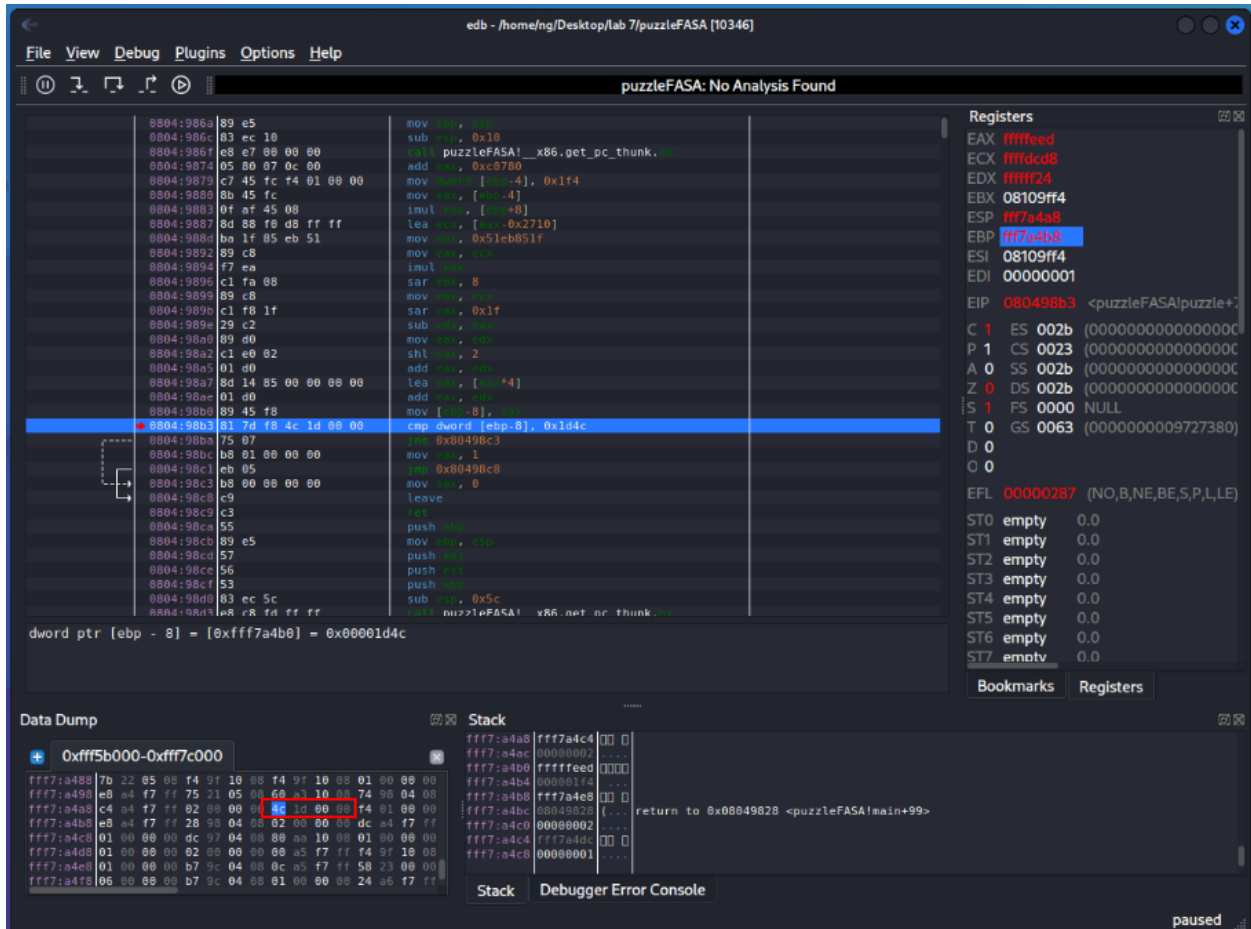
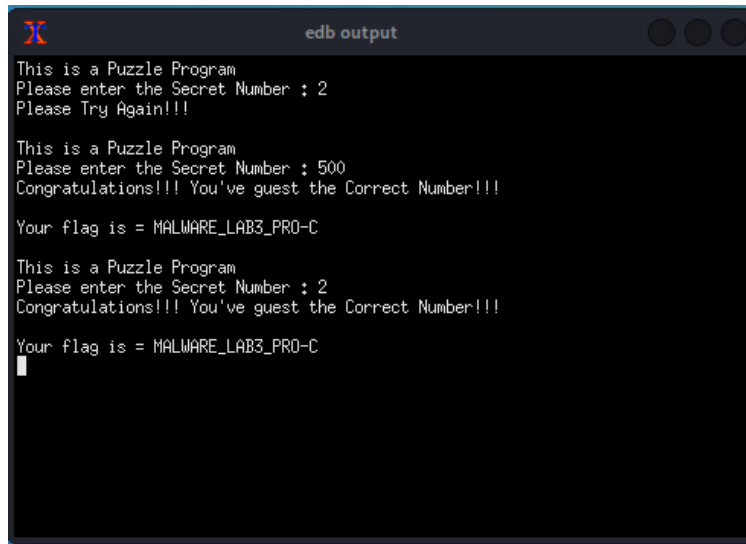


Figure 21

16. Run the puzzle program.

17. Program exits and shows flag even though the secret number entered is not the correct one.

(Figure 22)



```
edb output
This is a Puzzle Program
Please enter the Secret Number : 2
Please Try Again!!!

This is a Puzzle Program
Please enter the Secret Number : 500
Congratulations!!! You've guest the Correct Number!!!

Your flag is = MALWARE_LAB3_PRO-C

This is a Puzzle Program
Please enter the Secret Number : 2
Congratulations!!! You've guest the Correct Number!!!

Your flag is = MALWARE_LAB3_PRO-C
█
```

Figure 22

References

- 1) *Introduction to reverse engineering with ghidra*. Hackaday.io. (n.d.). Retrieved December 8, 2022, from <https://hackaday.io/course/172292-introduction-to-reverse-engineering-with-ghidra>
- 2) *X86 assembly guide*. Guide to x86 Assembly. (n.d.). Retrieved December 8, 2022, from <https://www.cs.virginia.edu/~evans/cs216/guides/x86.html>
- 3) *ARM – memory management unit - keil*. (n.d.). Retrieved December 8, 2022, from https://www.keil.com/dd/docs/datashts/atmel/at91sam9260_dc.pdf
- 4) *Decompiler Window*. Fossies. (n.d.). Retrieved December 8, 2022, from <https://fossies.org/linux/ghidra/Ghidra/Features/Decompiler/src/main/help/help/topics/DecompilePlugin/DecompilerWindow.html>