Practical Session – Week 8

Objectives

1. Interpret and manipulate assembly code via hardware debugging techniques
2. Apply reverse engineering techniques to identify main software flaws

Basic commands on linux

1. *objdump -d binary.file : show all assembly*
2. *strings binary.file : show all strings*
3. *gcc file.c -o binary.file -g -O0/3*
4. *gcc file.c -S assembly.s -O0/3*

Basic commands on gdb

Please see this link https://sourceware.org/gdb/current/onlinedocs/gdb/

1. set disassembly-flavor intel : show intel syntax instead of AT&T
2. break or b : set a break point
   - b main : break to main function
   - b *0x0342FA0230 : break to this program address
3. run : goes to the first breakpoint
4. continue : run/go to the next breakpoint
5. return : step out of the function by cancelling its execution
6. si : Execute one machine instruction, then stop and return to the debugger
7. x/s : show the content of specific memory address
   - x/s 0x402400 or x/s $rax
8. info registers or i r : show the content of the registers, e.g., i r $rip shows the next instruction to be executed (%rip register holds the next instruction)
9. disas : show the assembly code at this point, or use 'disas function1' to display the assembly of this function
10. print : display individual register value
    - print /d $rax : display the value of rax register in decimal
    - print /t $rax : display the value of rax register in binary
    - print /x $rax : display the value of rax register in hexadecimal
11. The "x" command is used to display the values of specific memory locations: "x/nyz"
    - "n" is the number of fields to display
    - "y" is the format of the output, 'c' for character, 'd' for decimal and 'x' for hexadecimal
    - "z" is the size of the field to be displayed, 'b' for byte, 'h' for 16-bit word, 'w' for 32-bit word
    - 'x/10xw $rsp' : displays in hex first 10 32-bit contents of the stack

Task – Bomb Lab, try to defuse the Bomb

This week you will reverse engineer the bomb lab game and try to defuse the bomb. This is an original source of exercise from Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective, Carnegie Melon University. You can find more information about this task in http://csapp.cs.cmu.edu/public/labs.html.

This game consists of 6 phases. In each phase, you must enter the right password otherwise the bomb explodes. ***In this practical, you will defuse just the first phase***.

How to run it: You can just type *'./bomb'* or type *'./bomb inputs.txt'*, where in '*inputs.txt*' the input strings are.

How to start defusing the bomb: First, you must use "*objdump –d bomb*" command to generate its assembly. You can use "*objdump –d bomb > output.txt*" command to redirect the output to a .txt file. You can also run the command "*strings bomb > output2.txt*" to see all the strings of the binary; a password might be stored there, which is a serious software flaw (actually it does). Part of the C-code is also provided in bomb.c file, to better understand the structure of the code. After you have had an idea of the program's functions, it is time to start reverse engineering phase1 routine (in this practical you will defuse only the first phase of the game). Can you identify where the input message is read? Use gdb and start studying phase1 routine step by step trying to understand what the code does. Make sure you understand what every instruction does.